NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

ONE SIZE DOES NOT FIT ALL: A SYSTEM DEVELOPMENT PERSPECTIVE

by

Erik LaSalle

September 2013

Thesis Advisor:	John Osumundson
Second Reader:	Kishore Sengupta

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2013	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE ONE SIZE DOES NOT FIT ALL: A SYSTEM DEVELOPMENT PERSPECTIVE		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Erik LaSalle			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. government. IRB protocol number ____N/A____.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE** A	

13. ABSTRACT (maximum 200 words)

Investments in technology have the potential to improve lives and organizations and can be force multipliers for an organization, however federal IT projects too often experience cost overruns, schedule slippages, and performance shortfalls. Specific to the Coast Guard, there are currently four Information Technology Level 1 acquisitions programs that have life-cycle costs estimates equal to or greater than $1-billion. Many of these projects are over budget, and as a result, many of the desired capabilities do not make it to the end user.

Since the passage of the first Acquisition Act and every acquisition mandate since, the federal government has struggled to deliver capabilities that have met the requirements of the end-user, while staying within budget, on schedule and within cost. To alleviate this, adding more mandates and oversight has become the "go to play." However, these mandates just might be having the antithesis effect on desired outcomes. This thesis describes alternative system development methodologies that could assist Department of Homeland Security and Department of Defense in maximizing the delivery of capabilities to the end-user, while staying on schedule and within budget.

14. SUBJECT TERMS WatchKeeper, MASI, agile, software development, project management, computer, engineering process, life-cycle, acquisition.	**15. NUMBER OF PAGES** 99
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

ONE SIZE DOES NOT FIT ALL: A SYSTEM DEVELOPMENT PERSPECTIVE

Erik LaSalle
Lieutenant Commander, United States Coast Guard
B.A., Florida Atlantic University, 2007

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
September 2013

Author: Erik LaSalle

Approved by: John Osmundson, PhD
Thesis Advisor

Kishore Sengupta, PhD
Second Reader

Dan Boger, PhD
Chair, Department of Information Sciences

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Investments in technology have the potential to improve lives and organizations and can be force multipliers for an organization, however federal IT projects too often experience cost overruns, schedule slippages, and performance shortfalls. Specific to the Coast Guard, there are currently four Information Technology Level 1 acquisitions programs that have life-cycle costs estimates equal to or greater than $1-billion. Many of these projects are over budget, and as a result, many of the desired capabilities will not make it to the end user.

Since the passage of the first Acquisition Act and every acquisition mandate since, the federal government has struggled to deliver capabilities that have met the requirements of the end-user, while staying within budget, on schedule and within cost. To alleviate this, adding more mandates and oversight has become the "go to play." However, these mandates just might be having the antithesis effect on desired outcomes. This thesis describes alternative system development methodologies that could assist Department of Homeland Security and Department of Defense in maximizing the delivery of capabilities to the end-user, while staying on schedule and within budget.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ALMIS	Aviation Logistics Management Information System
AOPs	Abstract of Operations
BFT	Blue Force Tracking
C4ISR	Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance
COI	Critical Operating Issues
DAA	Designated Approving Authority
DHS	Department of Homeland Security
DOG	Deployable Operational Group
DSDM	Dynamic Software Development Method
EVM	Earned Value Management
GAO	Government Accountability Office
IMS	Integrated Master Schedule
IOC	Interagency Operations Center
IOP	Interagency Operational Planning
IPT	Integrated Product Team
IVT	Integrated Vessel Targeting
JCIDS	Joint Capabilities Integration and Development System
KPP	Key Performance Parameters
MASI	Mission and Asset Scheduling Interface
MDA	Maritime Domain Awareness
MHSOPS	Maritime Homeland Security Operations
MISLE	Marine Information for Safety and Law Enforcement
MNS	Mission Needs Statement
MSAM	Major System Acquisition Manual
OM	Operations Monitoring
OPAREA	Operating Area
ORD	Operational Requirements Document
PM	Program Manager
PORD	Preliminary Operational Requirements Document

SAFE	Port Security and Accountability for Every Port
SCC	Sector Command Centers
SDLC	System Development Life Cycle
WBS	Work Breakdown Structure
XP	Extreme Programming

ACKNOWLEDGEMENTS

I would like to convey my sincerest appreciation to Dr. John Osmundson, Department of Information Sciences, Naval Postgraduate School, for his interest, time, and guidance during this endeavor. I would also like to thank Dr. Kishore Sengupta, Department of Information Sciences, Naval Postgraduate School for his tireless effort and passion in helping me write this thesis. Dr. Sengupta's understanding of the application of agile software development within project management is astounding, and without his guidance, I would have produced an inferior product. In addition, I would like to thank the Acquisition Research Program at the Naval Postgraduate School, specifically Ms. Tera Yoder for her support and guidance throughout the thesis process. I am grateful for her friendship and contagiously optimistic outlook on life.

Additional acknowledgements begin with Lieutenant Commander Christopher Treib of the United States Coast Guard. Chris's friendship and contributions to this endeavor is greatly appreciated.

I also would like to thank my beautiful wife, Dare, for her love, encouragement and support throughout my career and during my studies at Naval Postgraduate School. I value her friendship and love more than words can express, and without doubt, her tireless effort is the foundation of our family

Last, but certainly not least, I would like to thank my wonderful daughters, Isabella, Wylee, and Fenway, for being awesome kids and for reminding me on a daily basis of the true values of life

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Software Engineering

> The philosopher Hegel hypothesized that increased human understanding follows a path of thesis (this is why things happen the way they do); antithesis (the thesis fails in some important ways; here is a better explanation); and synthesis (the antithesis rejected too much of the original thesis; here is a hybrid that captures the best of both while avoiding their defects). (Boehm, 2006)

Statement of the Problem

The problem is that U.S. Coast Guard information technology (IT) projects are often delivered late, over budget, and not within the scope of the original requirements. Additionally, when these IT projects are delivered, they are often obsolete because the technology specified in the original acquisition requirements has a very short life cycle. This is a problem because failing to successfully deliver these IT capabilities hampers the Coast Guard's ability to accomplish its three primary responsibilities of maritime safety, maritime security, and maritime stewardship.

Purpose of the Study

The purpose of this thesis is to explore and understand the factors that may have contributed to Coast Guard IT projects that have been delivered late and/or out of scope or that are over budget. This study seeks an understanding of the nature and characteristics of failed IT projects. These failures are in the context of a plethora of resources made available to the Coast Guard to ensure the success of its IT projects. This study is important because it could identify several areas where progress might be made in improving the rate at which Coast Guard Command, Control, Computers, Communications, Intelligence, Surveillance, and Reconnaissance (C4ISR) technology can be assessed, acquired, implemented, and sustained.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Investments in technology have the potential to improve organizations and can be force multipliers. However, federal IT projects too often experience cost overruns, schedule slippages, and performance shortfalls. Specific to the Coast Guard, there are currently four information technology command, control, communications, computers, intelligence, surveillance, and reconnaissance (C4ISR) Level 1 acquisitions programs, which are outlined in the *Major Systems Acquisition Manual* as having life-cycle cost estimates equal to or greater than $1 billion. As stated in a September 2012 Government Accountability Office (GAO) report, these major C4ISR programs are 86 percent over budget and current funding levels will not allow the programs to execute as planned (Government Accountability Office [GAO], 2012). Additionally, outdated program baselines do not reflect current costs or schedules of the programs for myriad reasons, which results in the Coast Guard not being able to provide Congress with accurate information in its capital investment plan.

Since the passage of the first acquisition act, and in every acquisition mandate since, the federal government has struggled to deliver capabilities that have met the requirements of the end user while staying within budget and on schedule. To alleviate this challenge, adding more mandates and oversight has become the "go-to play." These policies and mandates, however, just might be creating a phenomenon that Senge (1990) called compensating feedback, which is "when well-intentioned interventions call forth responses from the system that offset the benefits of the intervention" (p. 58), meaning that the additional regulatory requirements are having a counterproductive effect on the desired outcomes. Regardless, improvements must be made, and this thesis explores a viable option for improvement.

A. ORGANIZATION OF THE STUDY

Chapter I of this thesis described the context surrounding the current state of investments in technology and C4IT capabilities within the Coast Guard. Chapter II describes agile software development and provides a glimpse into the current

fundamental application of this methodology. Chapter III provides a detailed look at both the WatchKeeper project and the Mission and Asset Scheduling Interface (MASI) project, two IT projects that I was personally involved with, and the outcomes of those projects. The goal of Chapter III is to provide a glimpse into the challenges that are present when fielding C4IT systems. Chapter IV discusses the challenge of information from federal-level policies and directions, as well as internal Coast Guard policies and direction. Chapter V presents potential considerations for future C4IT development endeavors.

B. MOTIVATION

I am convinced that the Coast Guard can become more efficient and effective at fielding capabilities for operators to be better positioned to complete their mission. Being involved with both the WatchKeeper project and the MASI project, I have witnessed firsthand successful outcomes to IT project management challenges—when the effort is freed of bureaucratic mandates that have little to no value. I am also convinced that the Coast Guard possesses enough indigenous talent to accomplish fielding useful systems for our operators.

C. RESEARCH QUESTIONS

1. Introduction

- What is the problem and purpose of the thesis? Agile development
- What is it?
- What are the different types?
- What are the strengths and weaknesses?
- When is it appropriate to apply the methodology?
- What are the comparisons with traditional engineering approaches?

2. WatchKeeper and MASI IT systems

- What is the WatchKeeper project, and what were the goals and objectives of the project?
- How was the WatchKeeper project managed?
- What was the outcome of the WatchKeeper project?

- What is the MASI project, and what were the goal and objectives of the project?
- How was the MASI project managed?
- What was the outcome of the MASI project?

3. Analysis of the WatchKeeper and MASI projects
4. Recommendations

THIS PAGE INTENTIONALLY LEFT BLANK

II. WHAT IS AGILE?

Agile software development is an approach that developers use to plan, coordinate, work, and communicate with customers, stakeholders, etc. In its most simplistic form, agile software development is about "feedback and change" (Dingsøyr, Dyba, & Moe, 2010). Cockburn (2006) also stated that by accepting that perfect communication is not feasible, one can learn to manage that uncertainty and "stop when you have sufficiently communicated to the purpose of the intended audience" (p. 1). Boehm and Turner (2004) defined *agile* as both the ability to rapidly change and the counterpart to discipline: discipline strengthens; agility releases and invents. A textbook definition of *agile development* states that when there are uncertainties with development or problems occur, agile provides procedures for allowing for flexibility to be responsive to unanticipated issues (Burd, Jackson, & Satzinger, 2012). Erickson, Lyytinen, and Siau (2005) defined *agility* as the "means for stripping away the heaviness, commonly associated with traditional software development methods, to promote quick response to changing environments" (p. 2). These definitions don't necessarily solidify an exact answer to what agile software development is; however, the definitions share some similar terminologies, such as communication, uncertainty, volatile environments, and flexibility—all of which are derived from the *Manifesto for Agile Software Development* (Beck et al., 2001).

A. MANIFESTO FOR AGILE SOFTWARE DEVELOPMENT

In February 2001, 17 people met in Utah and developed what is commonly known as the *Manifesto for Agile Software Development* (the *Manifesto*). The *Manifesto* describes what the group feels is "the uncovering of better ways to developing software" (Beck et al., 2001). The *Manifesto* has 12 principles:

1. Satisfy the customer through early and continuous delivery of valuable software: The highest priority of the team is to satisfy the customer with frequent deliveries that allow for early feedback with respect to the requirements, the team, and the process.
2. Harness change for competitive advantage: If the team can adapt to the changing requirements (because of early, frequent delivery), this

allows for a response to late-breaking information that often allows a company to outmaneuver a competitor.

3. Deliver working software frequently: This reinforces the importance of delivering working software frequently.
4. Business people and developers work together daily: This principle enforces the concept that daily interaction helps to facilitate better communication.
5. Build projects around motivating people: This principle focuses on the people aspect of the project more than on the process.
6. Face-to-face conversation is the most effective and efficient way to convey information: This principle supports number 4, with the addition of the importance of face-to-face communication—the most efficient and effective approach for conveying information.
7. Working software is the primary measure of progress: This is the Manifesto's third reference to the delivery of working software. It reinforces software delivery as a primary goal of a software development project.
8. Agile processes promote sustainable development: This principle focuses on the nonlinearity of humans and suggests that as people put in long hours, they begin to tire and the rate of progress of the project slows
9. Continuous attention to technical excellence and good design enhances agility: This principle focuses on a well-encapsulated design, which facilitates greater agility and an ability to change. In order to accomplish this, the team should produce good designs throughout the project.
10. Simplicity is the art of maximizing work done: Simplicity is essential. As Cockburn (2002) stated, "Simplicity has to do with accomplishing while not doing, maximizing the work not done while producing good software" (p. 212).
11. The best architecture, requirements, and design emerge from self-organizing teams: The focus here is on the architecture being allowed to adjust over time, just as the requirements do.
12. Adjust and fine tune the development process to become more effective at delivering useful code in intervals: This principle reaffirms that the most important aspect of the software development project is the delivery of working software.

The four core values gleaned from the *Manifesto* and that are at the core of agile system development are the following

1. individuals and interactions over processes and tools,
2. working software over comprehensive documentation,
3. customer collaboration over contract negotiation, and
4. responding to change over following a plan.

B. GENERAL GUIDELINES TO AGILE (BASIC THEORIES)

The idea of agile development is that it is more important to place emphasis on the people in the project than on the documentation. Amicability, talent, skill, and communication become the foundation of the team, and the development of these skills is of utmost importance (Cockburn & Highsmith, 2001b). The idea is that by strengthening these areas, the cost of moving information and quickening the decision-making sequence is realized, ultimately making the team more flexible. By placing people physically closer and replacing documents with in-person communication, the cost of moving information can be greatly reduced; likewise, adding experts to the team and working incrementally quickens the feedback loop, thus reducing the time that it takes to make a decision (Cockburn & Highsmith, 2001b).

The fact that the business world has become turbulent, uncertain, and fast paced—requiring fast responses—is why the term *agile* has been coined. However, it is of the utmost importance not only to be fast, but to be accurate as well. The agile process requires that appropriate business processes be in place to make and support change. However, in order for these processes to succeed, they must have responsive people and organizations. Too often, software engineering and rigorous process adherents are incorrectly confused as competence (Cockburn & Highsmith, 2001a). As Cockburn and Highsmith (2001b) stated, "Processes do provide the framework for groups to work together, but processes alone cannot overcome a lack of competency. However competency can surely overcome the vagaries of a process" (p. 132).

Agile software development is a complex phenomenon that includes interrelated practices and managerial policies, so it might be best to try to examine agile software development from a theoretical perspective. There are a variety of theories that best

explain agile development, but dynamic capabilities, coordination, and double-loop learning are the theories that work the best. Dynamic capabilities theory helps explain the need for agility and how to achieve it. Coordination and double-loop learning help explain how to best achieve coordination and learning in an agile environment (Balasubramaniam & Lan, 2007). Agile manufacturing, which was introduced to help the United States regain competitive positioning in the manufacturing world, proves that agility is not unique to software development (Dingsøyr & Dybå, 2008). Manufacturing industries embraced agile to react quickly to changing customer requirements, and dynamic capabilities theory, as explained in strategic management literature (Balasubramaniam & Lan, 2007). Pisano, Teece, and Teece (1997) stated, "Dynamic capabilities are the firm's ability to integrate, build and reconfigure internal and external competences to address rapidly changing environments" (p. 515). Dynamic capabilities theory explains how organizations can achieve competitive advantages while operating in a changing environment. Dynamic capabilities theory exhibits several common features of agile development. These features include cross-functional teams, joint experiences among team members, and external communications. Effective dynamic capabilities include the frequent use of prototyping to obtain real-time feedback in order to adjust actions and experimentations. Applying dynamic capabilities theory to agile software development has been proven successful across multiple industries, suggesting merit to its application in appropriate dynamic environments.

Coordination theory requires that the entire group working on the project share a common set of goals and share information to facilitate activities (Kraut & Streeter, 1995). As task interdependence becomes intensive, group coordination increases significantly and personal coordination increases moderately (Van De Ven, Delbecq, & Koenig, 1976). As such, agile development involves intensive teamwork and high task interdependence, using group meetings and personal coordination. As task interdependence increases, organizational hierarchy decreases, suggesting that agile development requires an increased use of organizational rules and routines (Balasubramaniam & Lan, 2007). As uncertainty increases, tasks become more challenging and coordination is more difficult. Therefore, the use of personal and group

coordination increases while the use of impersonal coordination decreases significantly. In an agile environment, where tasks are highly uncertain because of changing and/or incomplete requirements, personal and group modes of coordination are preferred over the use of formal documentation (Balasubramaniam & Lan, 2007). As Balasubramaniam and Lan (2007) state, "Agile approaches replace heavy documentation, upfront design, detailed plans and formal contracts with feature based planning, evolving design and co-located customers" (p. 46).

Double-loop learning theory helps explain how to solve complex and ill-structured problems in rapidly changing contexts. Learning is critical in agile software development, as Highsmith (1997) states,

> In an adaptive environment, learning challenges stakeholders, including both developers and customers, to examine their assumptions, and then use the results of each development cycle to learn the direction of the next. The cycles need to be short, so teams can learn from small, rather than large mistakes. They also need to be double-looped, so teams learn both about product changes, fundamental changes, and underlying assumptions about how the products are being developed. (p. 45)

Double-loop learning theory has three important elements:

1. Governing variables are dimensions that people keep within.
2. Action strategies are the plans used to keep governing variables within an acceptable range.
3. Consequences are the results of those actions.

In single-loop learning, when something goes wrong, workers try to look for another solution given the variables that are present. In double-loop learning, however, people question the governing variables themselves and subject the variables to critical scrutiny. As such, this is a shift in the way people frame strategies and consequences. Double-loop learning is more important for organizations operating in dynamic environments (Argyris & Schön, 1996). See Figure 1 for details on single- and double-loop learning.

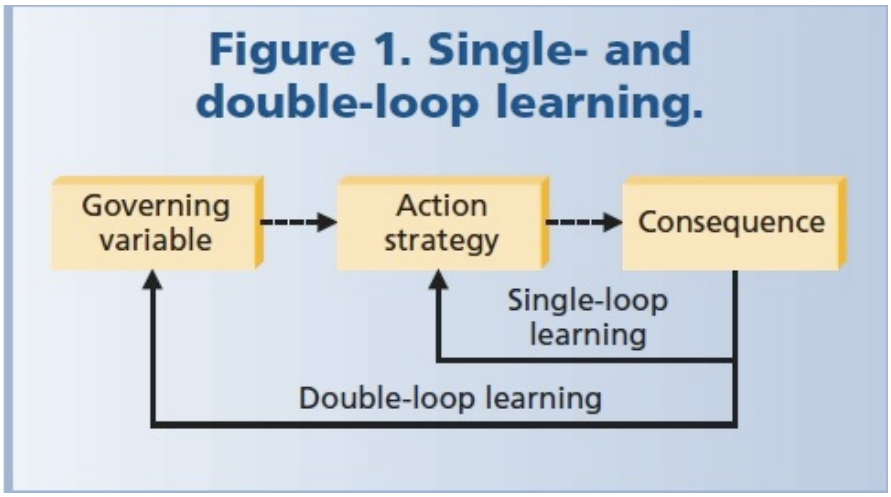

Figure 1. Single- and Double-Loop Learning (From Argyris & Schön, 1996)

Agile software development practices foster double-loop learning because double-loop learning provides an environment that warrants the participants to experiment with their mental models. As Beck (2004) stated, "Agile software development is a continuously self-correcting process" (p. 46). Instead of doing things right, the focus is on doing the right thing to enhance business value, frequently adjusting strategies and monitoring the feedback of those decisions.

An aspect of agile development that is often missed is that organizations are complex, adaptive systems, where decentralized interaction is guided by a set of simple, generative rules (Cockburn & Highsmith, 2001b). The previously mentioned organizational theories are tremendously important to help explain why agility is useful in software development. The agile approach is consistent with these sound principles and is grounded in management and the organization theories explained previously (Balasubramaniam & Lan, 2007).

C. AGILE DEVELOPMENT METHODS

Agile software development methods are being adopted in all industries and fields to deal with quickly evolving requirements that can become obsolete before project completion (Balasubramaniam & Lan, 2007). As Sengupta et al. (2013) stated:

> Lightweight processes that employ short iterative cycles, actively involve users to establish prioritize, and verify requirements, and rely on a team's tacit knowledge as opposed to documentation. True agile methods must take several cycles to complete, teams must determine the best way to handle work, and the work structures must be reorganized during the project rather than predetermined. (p. 2)

There are multiple agile methodologies. To provide a scope for this thesis, I summarize six methods in this section: crystal methodologies, dynamic software development method (DSDM), feature-driven development, lean software development, scrum, and extreme programming (XP). I do not intend for this section to describe these methodologies in great detail but rather to provide high-level exposure to each method's core values and practices (Dingsøyr & Dybå, 2010). Methods/processes/models are not capitalized in APA (like laws and theories are not).

1. Crystal Methodologies

The core philosophy of this methodology is that software development requires cooperative invention and communication, with a primary goal of delivering useful working code. A key to this philosophy is that projects need to be run differently based on needs and that the people involved must be as flexible as the needs. Crystal is a method for co-located teams of different sizes and criticality, and each team is given a color based on the team's size and the team's talents. These colors are clear, yellow, orange, red, magenta, and blue. As shown in Figure 2, the clear team has the fewest members, while the blue team has the largest number of team members. This is the most flexible of all the agile methods and critically focuses on communication and small teams (Cockburn, 2002). Crystal development has seven characteristics:

1. frequent delivery,
2. reflective improvement,
3. osmotic communication,
4. personal safety,
5. focus,
6. easy access to experts, and
7. requirements for the technical environment. .

	Clear	Yellow	Orange	Red	Magenta	Blue
Life	L6	L20	L40	L100	L200	L500
Essential	E6	E20	E40	E100	E200	E500
Discretionary	D6	D20	D40	D100	D200	D500
Comfort	C6	C20	C40	C100	C200	C500
	1-6	20	40	100	200	500

Figure 2. Agile Development Crystal Methodologies (From John Pruitt, 2011)

Figure 2 describes the various categories of the crystal method, of which life has the highest priority and comfort has the lowest. The colors represent the size of the team that is needed for the effort. For example, an E-yellow project is a project that is essential and requires a team of 20 members, and a D-red project is a project that is discretionary and requires a 100-person team.

2. Dynamic Software Development Method (DSDM)

This methodology divides projects into three phases: pre-project, which focuses on candidate projects and funding; project life cycle, which examines the feasibility, design, and implementation of the project; and finally, the post-project, which ensures the system is operating effectively and efficiently. Figure 3 provides a graphical representation of the DSDM. The DSDM has nine principles:

1. involving the user,
2. empowering the project team,
3. delivering frequently,
4. addressing current business needs,
5. using iterative and incremental development,
6. allowing for revisions,
7. fixing high-level scope before the project starts,
8. testing throughout the project life cycle, and
9. providing efficient and effective communication.

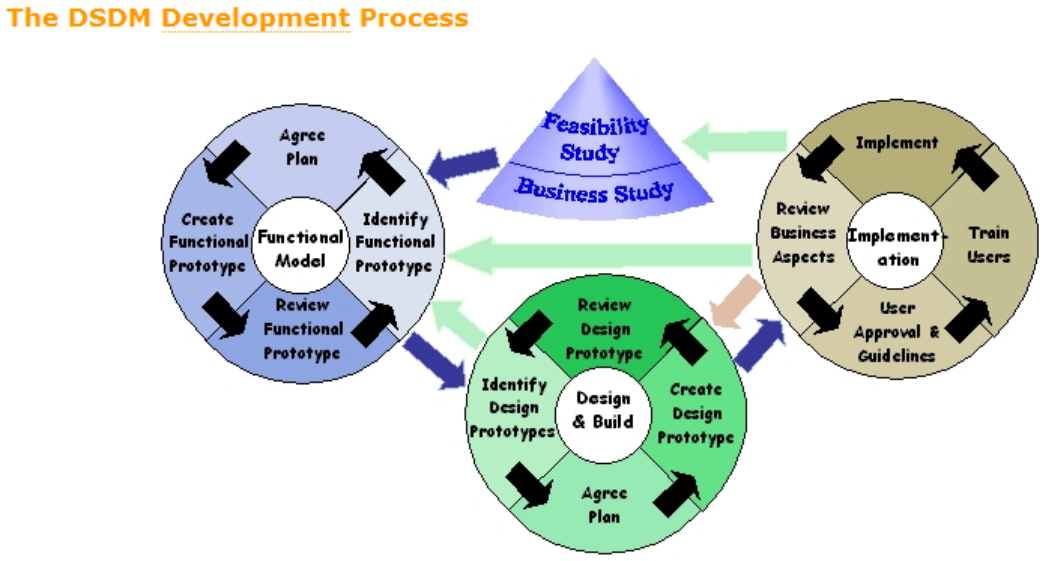

Figure 3. Dynamic Development Software Method (From Clifton & Dunlop, 2003)

3. Feature-Driven Development

This methodology combines model-driven and agile development with an emphasis on an initial object model, division of work features, and iterative design for each feature. It consists of five activities: develop overall model, build feature list, plan by feature, design by feature, and finally build by feature. Feature-driven development is driven from the customer's perspective and is designed around industry best practices. Figure 4 provides a simple graphical representation of the feature-driven development model.

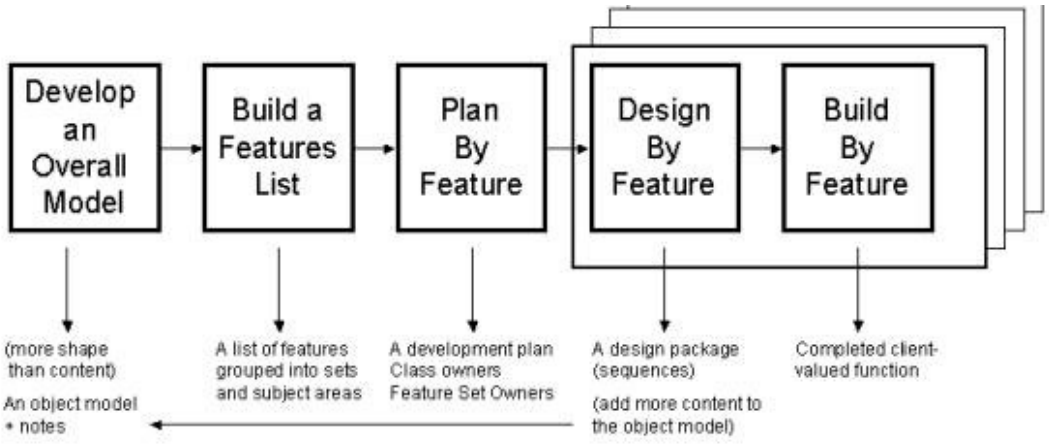

Figure 4. Feature-Driven Development (From Feature-Driven, n.d.)

4. Lean Software Development

This methodology is an adaptation of principles from lean production, in particular, the Toyota production system, to software development. This methodology has seven principles:

- eliminate waste,
- amplify learning,
- decide as late as possible,
- deliver as fast as possible,
- empower the team,
- build integrity, and
- see the whole.

Figure 5 shows a graphical overview of this method.

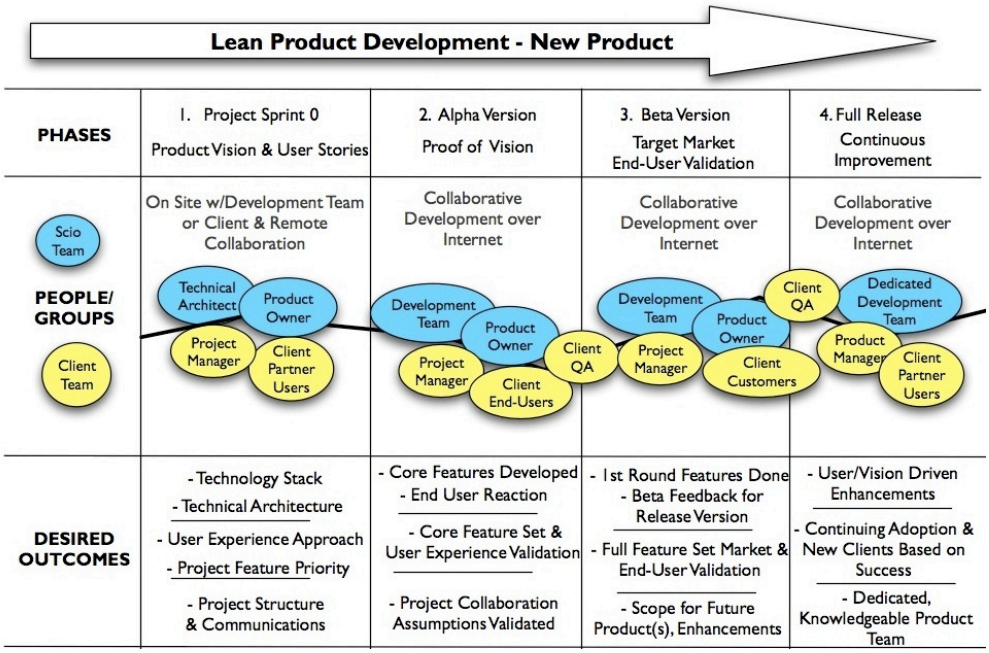

Figure 5. Lean Software Development (From Scio, 2010)

5. Scrum

This methodology focuses on project management in situations where it is difficult to plan ahead and where feedback loops constitute the core element of the process. Software is developed by a team in increments that are called sprints, starting with planning and ending with review. The implementation features are registered in a backlog, and the product owner decides which backlog items should be developed in the next sprint. All of the software development activities (requirements analysis, design, coding, testing, and delivery) are carried out in each sprint (Suganya & Mary, 2010). At the end of each sprint, the team is able to deliver a small portion of the product. Work is coordinated in daily stand-up meetings where the person in charge, called the scrum master, is responsible for solving problems. These scrums define the framework to organize and produce products on time. The scrum master prioritizes the backlog, and then the scrum team prioritizes the customer requirements, taking into consideration both the customer needs and the business needs. Figure 6 shows a graphical representation of a scrum and the tasks involved.

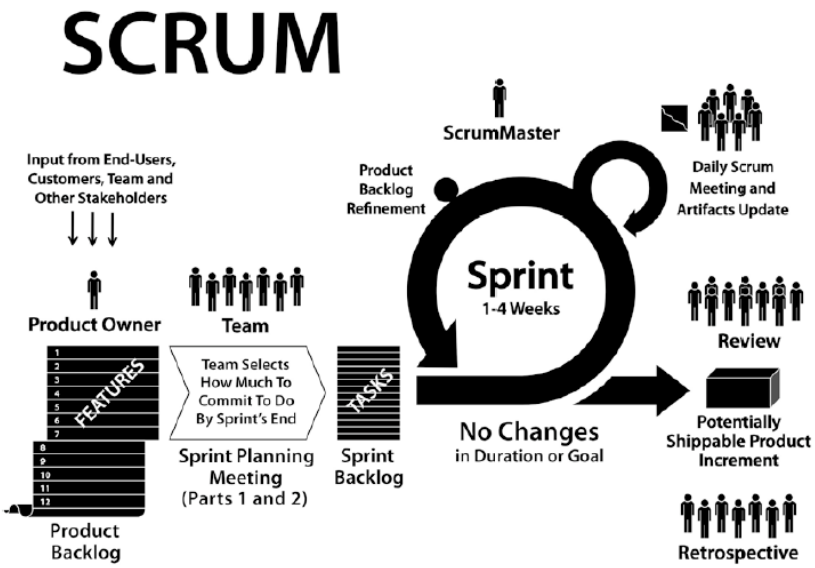

Figure 6. Scrum Development (From Lynch, 2010)

6. Extreme Programming (XP)

This methodology is probably the most well-known agile process (Beck, 2000; Strigel, 2001). XP starts with a planning phase, followed by several iterations, and ends with acceptance testing. The work is broken up and prioritized by the end user. The key is that at the end of every iteration, the end user performs an acceptance test against the requirements, often referred to as user stories (Suganya & Mary, 2010). See Figure 7 for a graphical depiction of the XP process. XP focuses on best practices for development and consists of 13 common practices:

- whole team,
- customer test,
- small releases,
- planning game,
- collective ownership,
- coding standard,
- continuous integration,
- metaphor,
- sustainable pace,
- simple design,

- pair programming,
- refectory, and
- test-driven development.

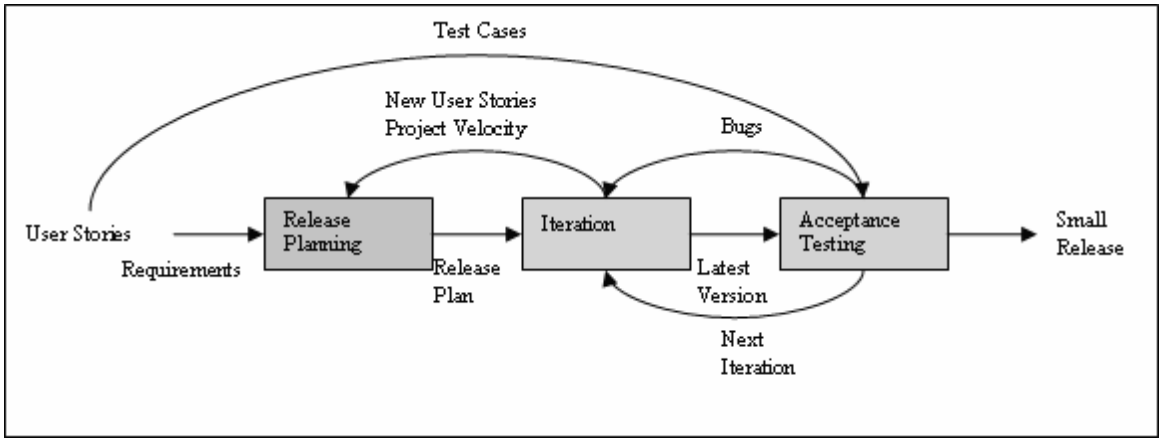

Figure 7. Extreme Programming (From Extreme, 2000)

Given the preceding examination of these agile methodologies, there are six features that are common to them all. As Bohner and Coram (2005) stated, these features are as follows:

1. Collaboration: Agile methods are highly collaborative inside and outside the development group.
2. Code review: Agile methods encourage code reviews to facilitate the dissemination of key information.
3. Small teams: Agile methods encourage small teams.
4. Short release schedules: Agile release schedules can be as short as two weeks, which allows the team to evaluate the product and identify priorities.
5. Boxing: Time boxing helps to focus the customer and reduces scope creep. The release length is fixed so that the features of the system are not.
6. Constant testing: Frequent testing helps to prevent a degraded product. This helps to offset the risk of just writing the code. Testing must be automated with the daily builds and regression test to ensure that all functionality works.

To support the theme of common characteristics, Glaiel, Moulton, and Madnick (2013) have found that regardless of the agile methodology employed, seven agile

techniques are common to all of the previously mentioned methodologies. They call these seven agile techniques the "Genome of Agile," and they are listed and described in Table 1 (Glaiel, Moulton, and Madnick, 2013).

Table 1. The Genome of Agile (From Glaiel et al., 2013)

1. Gene Name	2. Description	3. Contrast to Traditional
Story/Feature Driven	Breakup of the project into manageable pieces of functionality, sometimes named "features," "stories," "use cases," or "threads." The system is segmented into sets of client-valued functionality, and development work is organized around producing these features.	Traditionally employ functional decomposition where system is broken into subcomponents that are implemented in parallel and integrated in late stages. This requires upfront requirement specification in lockdown.
Iterative-Incremental	Development is performed in repeated cycles (iterative) and in portions at a time (incremental).	Development approaches call for complete requirements analysis phase, followed by lengthy design, coding, and test phases
Micro-Optimizing	This represents the adaptive nature of agile management processes. Agile methodologies are encouraged to tailor aspects of the development process to adapt to change. Teams are empowered to modify aspects of the process or dynamically adapt to changing circumstances. Small improvements and variable changes are made frequently as needed.	Traditional processes can exhibit a flavor of this change in the form of lessons learned activities that are called for at the completion of a project, but which really feed to the next development cycle and yield little improvement on subsequent development projects.
Refactoring	Refinement of the software design and architecture to improve software maintainability and flexibility. Several of the agile methodologies consider refactoring to be the primary	Typically traditional development schedules do not permit refactoring.

1. Gene Name	2. Description	3. Contrast to Traditional
	development practice. Refactoring consists of taking apart existing working code, factoring out common elements, and rebuilding it to provide a stronger base for subsequent development.	
. Continuous Integration	. Policies and practices related to configuration management, and software build and test automation. Continuous integration involves methods for maintaining an updated code base that includes all changes that have been made and regularly building a testable version of the product	. Configuration management is traditionally approached by having different teams develop different portions of software in isolated environments. They then try to integrate these separate portions later in the development cycle.
. Team Dynamics	. Soft factors related to the project team. Daily meetings, workspaces, pair programming, schedule/peer pressure, experience gained, etc.	.
. Customer Involvement	. Customer/User involved in demonstrations of functionality to verify and validate features. Higher frequency feedback and clarification of uncertainty. Availability to participate in development meetings. Customer involvement gene means accepting changing requirements and including the user in the development to the degree that is possible.	. The traditional approach to this is to lock in the system requirements early in the project. Any subsequent changes require contractual renegotiation for added scope or scope change.

The application of the genome of agile framework is dependent on the agile methodology used. Not every agile method features every genome as identified by Glaiel et al. (2013), as is shown in Table 2

Table 2. Agile Genes Maps to Several Popular Agile Methodologies (From Glaiel et al., 2013)

. Methodology	**26. Agile Gene**						
27.	28. Feature Driven	29. Iterative-Incremental	30. Refactoring	31. Micro-Optimizing	32. Customer Involvement	33. Team Dynamics	34. Continuous Integration
35. Scrum	36. X	37. X	38.	39. X	40. X	41. X	42.
43. XP	44. X	45. X	46. X	47. X	48. X	49. X	50.
51. TDD	52. X	53. X	54. X	55.	56.	57.	58. X
59. FDD	60. X	61. X	62.	63.	64.	65.	66.
67. Crystal	68. X	69. X	70.	71. X	72. X	73.	74.

D. ASSUMPTIONS AND IMPACTS

1. Assumptions

While there has been a lot of interest and enthusiasm behind agile methods, and most reviews have been favorable, specific assumptions are present in agile software development processes. These assumptions and development practices could lead to limitations. The following is a summary of the assumptions identified by Turk, France, and Rumpe (2005):

1. Visibility assumption: This assumption suggests that working code can be used as a sole source for project visibility. Although project visibility is traditionally accomplished through various report specifications—and measures of quality and productivity—agile development suggests that working code is a true barometer for project status.
2. Iteration assumption: This assumption states that a project can always be structured into short fixed-time iterations. As stated previously, agile processes require features to be coupled and bundled so they can be addressed in fixed-time iterations.
3. Customer-interaction assumptions: This assumption suggests that the customer will always be available for interaction when needed by developers. This means that the customer can always reschedule their other work.
4. Team-communication assumption: This assumption states that developers are located so that they are able to have frequent communication with each other, specifically face to face. This requires that team meetings be a priority and that this is accepted by all of the respective stakeholders.
5. Face-to-face assumption: This assumption suggests that face-to-face interaction is the most productive method in communication. This assumption deemphasizes the value of documentation as a communication aid based on the idea that tacit knowledge is superior to other types of gained knowledge. There are potential ramifications for this assumption. As Boehm (2002) stated, "This focus on tacit knowledge makes projects that use agile process dependent upon experts" (p. 13).
6. Documentation assumption: This assumption states that developing extensive documentation and software models is counterproductive. The assumption is that it is more reliable to determine design from actual code than from documents, specifically since documents are

rarely kept up to date and are not maintained when code is changed. Advocates for documentation state that documents provide good models to bring new hires up to speed, which helps users determine the applicability of requirements.

7. Welcoming changing requirements: Requirements change during software development, and this is recognized both in the agile and traditional developmental communities. Evolving requirements are an inherent problem of software development; however, it is assumed that the development team will be able to handle changing requirements, even late in the game.
8. Continuous redesign assumption: This assumption maintains that systems can continuously be redesigned while maintaining their conceptual integrity. The assumption is that the system can be redesigned and carried out without a significant amount of time and cost.
9. Simplicity is essential: This assumption states that the complexity imposed by heavyweight processes and models is unnecessary. The assumption is that a focused architecture that satisfies the current needs is preferred to a general architecture that is designed to incorporate future needs.

2. Impacts

In addition to the assumptions underlying agile software development processes, there are impacts that may affect the project management component of the software development effort. I examine these impacts as they relate to people, processes, and projects and then summarize the findings of Bohner and Coram (2005). The impact of a software development process on people is obvious. The people involved include developers, customers, testers, executive management, and project leaders, to name a few. However, the largest impact is on the developers. As previously stated, agile methods are lightweight methods that do not follow strict guidelines and processes. As such, it is imperative that the developers be highly trained and willing to work as a team. Cockburn (2002) identified characteristics and three levels of skill that developers must have to accomplish various tasks within a given framework. Table 3 identifies these characteristics.

Table 3. Levels of Software Understanding and Use (From Boehm & Turner, 2004)

Level	**Characteristics**
3	Able to produce solutions and unprecedented situations
2	Able to tailor solutions to fit new, but precedented situations
1A	Solid developer able to implement functionality, estimate effort, and re-factor code
1B	Able to implement simple functionality, execute tests, and follow directions
-1	Unwilling or unable to work in a collaborative environment

Of the three different personal technical skills identified in Table 3, only levels 3, 2, and 1A would possess the needed ability to work in an agile environment. Given the need to employ high levels of expertise, traditionally staffed organizations may have difficulties achieving this requirement.

The impact of using agile methodology on an organization's software development testing team is dependent on the developmental cycles of the agile process chosen. Testers must work closely with developers throughout the entire process and might actually need to be programmers themselves. The challenge to management is to be able to identify this required skill set of would-be team members. As a project leader in an agile development effort, the challenge is in assembling an experienced staff and empowering those members. This empowerment might be a cultural shift for some organizations, which may dissect the decision-making hierarchy. Additionally, project leaders have to develop the skills required to respond to change. Project leaders have a much more hands-on role than in traditional development efforts, and as such, they are more involved with customer collaboration.

Customers have a much more involved role with agile methods than with traditional development efforts. With agile, customers are involved throughout the entire process, unlike traditional development, where customers are involved only with defining the requirements and with acceptance testing. It is highly recommended that a full-time customer presence be on-site to work with the development team on a daily basis.

3. Process and Documentation Impacts

Since agile methods require new process activities, many organizations must make drastic changes to old processes to accommodate the new way of doing business. This includes, but is not limited to, planning, documentation, development processes, and delivery. Agile processes place less importance on formal planning, but planning still needs to take place. Planning in agile is a relatively informal process, but there are so many small tests, which may lead to more planning needs. In most agile efforts, documentation is often limited to allow for optional architecture to be developed. The determination of how much documentation to use in an agile effort is critical, as is the understanding that documentation must be updated whenever a change is made. Although this type of documentation effort can avoid the wasted time of writing a document and then leaving it to become obsolete, it does come with risk. As stated earlier, documentation is an excellent way to bring new hires up to speed with the developmental effort, and it provides a method for tracking and auditing.

4. Comparison to Traditional Engineering Methods (Plan Driven)

As stated previously, the primary goals of agile methods are rapid value and responsiveness to change, while the primary goals of plan-driven methods are predictability, stability, and high assurance. Agile approaches are based on the view that organizations are complex adaptive systems, where requirements are emergent rather than pre-specifiable (Boehm & Turner, 2004). Plan-driven goals are focused on increasing process capability for standardization, measurement, and control. Agile projects focus on building things quickly and finding out through experience what activity or feature will add the most value (Boehm & Turner, 2004). Agile methodologies are reactive postures that have considerable advantages when operating in an environment with rapid changes, such as technology. However, the downside to this approach is the overemphasis on tactical objectives over strategic objectives.

Current research has stated that agile processes work best within small to medium groups working on relatively small applications (Boehm & Turner, 2004). Kent Beck (2004) stated, “The size of the project clearly matters, and it would very difficult to run a

project using agile methods with a team of 100 programmers or more" (p. 38). Larger agile projects with hundreds of people have been successful, but in those cases, traditional plans and specifications were adopted to deal with interactions among the project elements. Conversely, plan-driven methods are better for larger projects, where plans, documentation, and processes provide for better communication. As stated previously, agile methods concentrate on delivering a product on time to satisfy the customer. However, this comes with an inherent risk of microscopically focusing on the product at hand and ignoring problems that may occur later. An example of this pitfall is developing an agile system or application that doesn't integrate well with the organization's overall enterprise architecture.

When agile approaches are compared to traditional approaches from a managerial perspective, there are discrete differences in each stakeholder's expectations. Although planning, control, and communications are prevalent in both approaches, they are managed differently. Agile approaches depend on dedicated customer involvement focused on adding rapid value to the effort. Conversely, plan-driven methods depend on a formal contract between the developers and customers as the basis for customer relations. This contract is designed to identify foreseeable problems in advance and formalize a solution with documentation. Although this approach aids in identifying potential issues, it can be a high stress point for the development team working to facilitate the plan-driven effort. With agile, planning is seen as a means to an end, and a high percentage of time is spent on re-planning. Plan-driven methods use plans to anchor their processes and again to provide for a spectrum for communication. As stated in the *Manifesto for Agile Software Development*, the emphasis in agile methodologies is on individuals and interactions (Beck et al., 2005).

An important part of agile development—maybe even the most important part—is testing. Testing is a way to validate that the customers have specified the right product and that the developers built the right product. Testing requires that code be developed and executed, which is frequent in agile approaches. However, with plan-driven approaches, testing does not occur as often, resulting in problems being discovered late in

the development cycle; these problems are expensive to fix. In most agile approaches, it is recommended to automate testing procedures. As Beck (2003) stated:

> This has significant advantages:
>
> - Ensures that the requirement is testable
> - Avoids documentation minutia
> - It enables incremental build and test opportunities
> - It helps modularize the application structure and provides a safety net for re-factoring
> - It helps form an explicit working knowledge of the application. (p. 74)

Table 4. Traditional versus Agile Software Development (From Nerur, Mahapatra, & Mangalara, 2005, p. 75)

	Traditional	**Agile**
Fundamental assumptions	Systems are fully specifiable, predictable, and can be built through meticulous and extensive planning.	High-quality, adaptive software can be developed by small teams using the principles of continuous design improvement in testing based on rapid feedback and change.
Control	Process centric	People centric
Management style	Command and control	Leadership in collaboration
Knowledge management	Explicit	Tacit
Role assignment	Individual: favors specialization	Self-organizing teams: encourages role interchangeability
Communication	Formal	Informal
Customer's role	Important	Critical
Project cycle	Guided by tasks or activities	Guided by product features
Development model	Life-cycle model (waterfall, spiral, or some variation)	The evolutionary delivery model
Desired organizational form/structure	Mechanistic (bureaucratic with high formalization)	Organic (flexible in part to dissipate encouraging cooperative social action)
Technology	No restrictions	Favors object-oriented technology

5. When to Apply Agile Development

Agile methodologies are appropriate for projects that have high variability, uncertain requirements, and unknown capabilities of people and that are utilizing new technology (Nerur et al., 2005). To better guide the decision-making requirements on when to use agile approaches and when to use plan-driven approaches, I have identified five critical factors, introduced by Cockburn et al. (2005) to be most appropriate. As described by Boehm and Turner (2004), these factors are project size, criticality, dynamism, personnel, and cultural factors. Table 5 describes these factors.

Table 5. The Five Critical Agility/Plan-Driven Factors (From Boehm & Turner, 2004, p. 55)

Factor	**Agility Discriminators**	**Plan-Driven Discriminators**
Size (number of personnel)	Well matched to small products and teams. Reliance on tacit knowledge limits scalability.	Methods evolved to handle large products and teams. Hard to tailor down to small projects.
Criticality (Loss due to impact of defect)	Untested on safety-critical products. Potential difficulties with simple design and lack of documentation.	Methods evolved to handle highly critical products. Hard to tailor down to low criticality products.
Dynamism (Percentage of requirements changing per month)	Simple design and continuous re-factoring are excellent for highly dynamic environments but a source of potentially expensive rework for highly stable environments.	Detailed plans and big design up-front are excellent for a highly stable environment, but a source of expensive rework for highly dynamic environments.
Personnel (Technical abilities as defined in Table 3)	Requires continuous presence of critical mass of scarce Level 2 or 3 experts (as defined earlier). Risky to use non-agile Level 1B people.	Need for critical mass of scarce Level 2 and 3 experts (defined earlier) during project definition but can work with fewer late in the project. Can usually accommodate some Level 1B people.

Factor	Agility Discriminators	Plan-Driven Discriminators
Culture (thriving in chaos vs. order)	Thrives in a culture where people feel comfortable and powered by having many degrees of freedom (thriving on chaos).	Thrives in a culture where people feel comfortable and empowered by having their roles defined by clear policies and procedures (thriving on order).

Boehm and Turner (2004) developed Figure 8, which does a nice job summarizing graphically the five critical factors associated with agile and plan-driven efforts. The closer you move towards the center of the diagram, the more appropriate it is to use agile methods. By rating a project along all of the five axes, a visual evaluation of relationships can be identified.

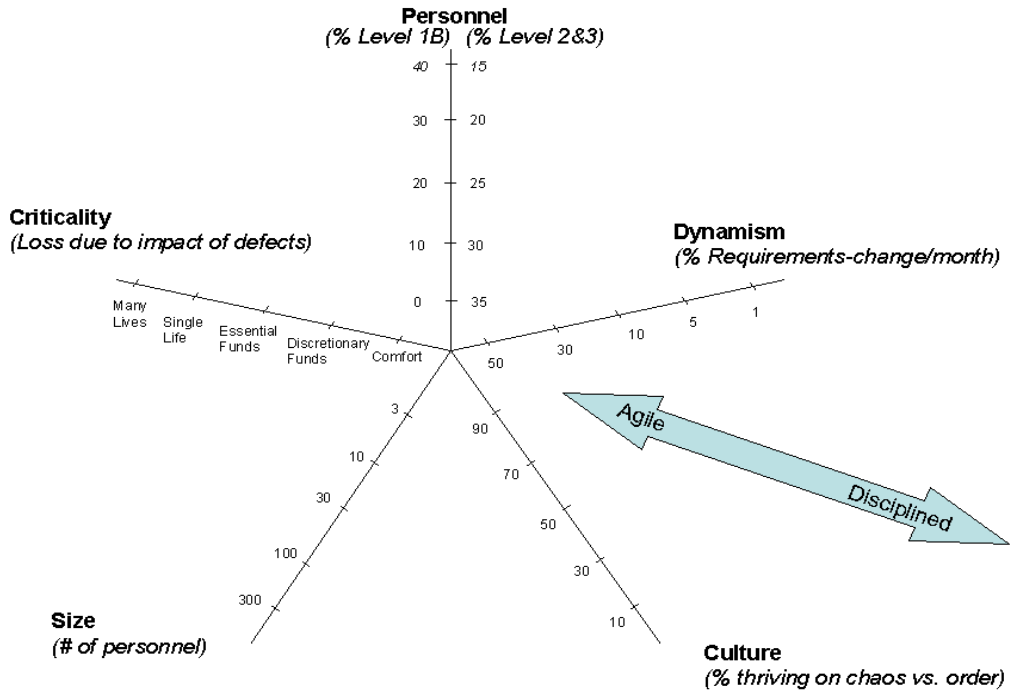

Figure 8. Dimensions Affecting Method Selection (From Boehm & Turner, 2004)

III. WATCHKEEPER AND MASI

As stated in Chapter I, my interest in agile system development started during my time working on both the MASI and WatchKeeper projects. The work I did on these projects came during my time while stationed at Coast Guard Headquarters from 2008 to 2011. My role with both projects was in the capacity of sponsor's representative, and, as such, my responsibilities included acting as a liaison between the end users of the systems (operators) and the rest of the project team. As the sponsor's representative, I worked very closely with the sponsor, and my main obligation was to ensure that requirements that the end user identified as important were built into the system being developed. In the following paragraphs, I will provide a synopsis of both the WatchKeeper and MASI projects. The projects will be broken down by their goals and objectives, the doctrine in process that was followed for each project, how progress was measured, the stakeholders within the projects, the communication effort within the projects, and finally the other factors that influenced the projects.

A. WATCHKEEPER GOALS AND OBJECTIVES

The WatchKeeper project was the IT component of the larger Interagency Operations Centers (IOC) project. The IOC project was a result of the mandates of the Security and Accountability for Every Port (SAFE Port) Act of 2006, and directed the Department of Homeland Security (DHS) to transform Coast Guard Sector Command Centers (SCCs) to host interagency members and meet the challenges of interagency coordination and maritime security. The three gaps identified by the SAFE Port Act were the following:

1. basic awareness of vessel activities near vulnerable port and coastal infrastructure,
2. systems linking the ever-increasing volume of information with vessels in ways that help decision-makers determine threats and develop the correct course of action, and
3. means for effective information sharing and joint operations with port partners.

The volume of maritime domain awareness (MDA) information necessary to manage Coast Guard and interagency operations has increased dramatically and exceeded the field's capacity to collect and process it. The Coast Guard needed new information management capabilities to solve the coordination and operational challenges faced by today's interagency decision-makers. Decision-makers lack the ability to see, understand, and share information that is critical to coordinate interagency operations in port and coastal areas. This situation severely inhibits efficient information sharing with interagency partners, resulting in reduced mission capabilities in the ports and waterways within the U.S. The WatchKeeper project was the IT system that was identified to help close these gaps, and as such, was targeted to provide the following capabilities:

1. Integrated vessel targeting (IVT): This component integrates the targeting results of various agencies, and builds a consolidated threat picture of people, vessels, and cargo operating within an operating area (OPAREA) as provided by intelligence and law enforcement communities in support of the Ports, Waterways, and Coastal Security missions.
2. Interagency operational planning (IOP): This planning component integrates federal, state, and local asset status and schedules. As such, better coordination and more efficient resource allocation between agencies can be realized.
3. Operations monitoring (OM): This component manages the IOC daily schedule that was created by the IOP component. It manages the schedule against all emergent events, such as search and rescue, spills, and other events occurring outside the operational planning window. OM creates and shares the tactical picture, including command and control, mission status, and the status of IOC forces and Blue Force Tracks (BFT).

B. WATCHKEEPER PROJECT PROCESS AND DOCTRINE

As the sponsor's representative for the project, I along with the sponsor's directorate was responsible for providing a few key documents early on in the project. These documents included the *mission need statement* (MNS), the Preliminary Operational Requirements Document, (pORD), and the Operational Requirements Document (ORD). The WatchKeeper project's MNS was approved in 2005 and was revalidated in February 2009. This MNS verified the capability gaps identified in the SAFE Port Act within ports and waterways within the U.S., and was used initially to

guide the needs of the project. In addition to the MNS, the pORD was developed in April 2008 to provide more fidelity to the actual requirements that would be needed, and to aid in the development of the more robust ORD, which was signed in 2010. Therefore, the only requirements guide that was used early on for the initial development effort was the pORD, which again, only provided a very high-level conceptual need, not system specific requirements.

For the project management effort, the following doctrine and Integrated Product Teams (IPTs) were used to guide the project:

- *Major Systems Acquisition Manual* (MSAM)
- *Systems Development Life Cycle* (SDLC*)*
- *Mission Engineering Process Guide:* This process guide was developed while working with and visiting select sectors to identify key processes and workflows of SCCs. The focus on the mission engineering effort was to identify processes and not IT solutions; the effort was to capture what was happening within the command centers at the various watch positions, to better define systems requirements.
- Two-chartered IPT: The two teams were
    1. information management IPT and
    2. senior leadership.

C. WATCHKEEPER PROJECT PROGRESS MEASUREMENT

One key MSAM requirement was that earned value management (EVM) be used as the performance measurement tool, because of the dollar threshold that WatchKeeper met. The goal of EVM is to integrate the contract scope of work with scheduling cost elements at appropriate levels for optimum project planning and control. The MSAM directs that EVM be used against a work breakdown structure (WBS) at sufficient levels to enable an understanding of the performance against the allocated time and budget. This information is then used to create an integrated master schedule (IMS), which incorporates the WBS items.

EVM is also the technique that communicates a project status within a portfolio and is an integral component of the Office of Management and Budget Exhibit 300

(Primavera Systems, 2008). The MSAM does not provide clear guidance on how EVM is to be incorporated but instead directs the PM to comply with the DHS (2009) guidance. The DHS guidance states:

> Title V of the Federal Acquisition Streamlining Act of 1994 requires agency heads to approve or define the costs, performance, and schedule goals for major acquisitions to achieve, on average, 90 percent of the cost, performance and schedule goals established (p. 8).

Additionally, when EVM is employed for a project, it is imperative that it be supported by management and stakeholders at all levels (Fleming & Koppelman, 2009). All stakeholders have a vested interest in the project, and it is important that everyone have a rudimentary understanding of the EVM data. EVM also enables stakeholders to understand what other stakeholders are doing. The following 10 requirements were identified by Fleming and Koppelman (2009) as being critical to successfully implement EVM:

1. EVM requires the project to be fully understood, defined, and scoped to 100 percent of the project effort. Stakeholders need to know what constitutes 100 percent of the work in order to measure progress along the way.
2. EVM requires that the defined scope be decomposed—broken down into major management tasks, which are selected as points of management control—and then planned and scheduled down to the detailed work package level.
3. EVM requires that an integrated and measurable project baseline be authorized—relating the scope of work directly to an achievable budget—then locked into a specific time frame for performance measurement. This is called bottom-up planning.
4. EVM requires that only authorized budgeted work be accomplished, meaning all work being done must be tightly controlled. Scope creep cannot be allowed.
5. EVM requires that physical performance be measured.
6. EVM requires that the values used be related to the planned values to accurately reflect performance against the project baseline.
7. EVM requires that reporting be consistent with the earned value being measured to allow for an accurate portrayal of cost performance. The relationship of actual cost must reflect the true cost performance. Earned value less actual cost provides cost performance.

8. EVM requires that a forecast be made periodically (weekly, monthly) to estimate the amount of time and money it will take to complete 100 percent of the project.

9. EVM requires that a full disclosure of actual results be made available to all stakeholders who have a vested interest in the project. All stakeholders will receive the same actual performance results.

10. EVM requires that project managers, in conjunction with key stakeholders, decide on the appropriate action to be taken to stay within authorized budget expectations.

D. STAKEHOLDERS, ROLES, AND RESPONSIBILITIES

Figure 9 at the end of this section provides a graphical representation of the hierarchy and organizational layout of WatchKeeper's stakeholders. It is not an official organizational hierarchy of the Coast Guard, but rather the organizational hierarchy of the WatchKeeper project from personal experience.

1. Sponsor and Sponsor's Representative

- Initially, CG-741 was the sponsor's representative and CG-761 was the sponsor. This later changed: CG-761 became a sponsor's representative and CG-741 maintained the role of the sponsor. The switch in responsibilities occurred when new leadership reported aboard both CG-761 and CG-741, creating a leadership turnover in both directorates. Both incoming captains were newly promoted, and no relationship between the two had been established yet. After the change in leadership, CG-761 took over the role of sponsor's representative and CG-741 assumed the duties of sponsor. Interestingly, the outgoing captain of CG-761 became the leader of the Command Control and Communications Center (C2CEN), which was later identified as the lead developer of the WatchKeeper system, and the outgoing captain of CG-741 retired.

- CG-9: Program manager: The program manager (PM) was a senior commander (O5) and had the overall responsibility of the project. This created an interesting dynamic in the senior decision-making for the project. Although the commander was more than capable of performing the duties required of the PM, there was still an underlying reality that he was junior to other decision-making stakeholders, given that they were all O6s.

- CG-6: CG-6 included the technical agents and technical leaders of the project. They were to oversee all engineering efforts with respect to impacts to enterprise, security, and accreditation. Both centers of excellence (C2CEN, Operations System Center [OSC]) are under CG-6

leadership; yet, for this development effort, the PM (CG-9) had the authority to direct the developers. This created an interesting dynamic in which the normal reporting and tasking chain of command was then bypassed; CG-9 directly tasked a CG-6 asset.

- C2CEN: When C2CEN was given the decision on which technical organization/corporation/agency to hire, the challenge was whether they should include themselves as a possible candidate for the job. After consideration, the decision was made that C2CEN would be the lead developing agents for this project.
- Operations Systems Center (OSC): OSC is another Coast Guard Center of Excellence that works for CG-6. Once it was decided that the Coast Guard was going to undertake this project in-house (from a developmental standpoint), OSC was earmarked for providing a piece of the proposed technical solution. As such, C2CEN would develop two thirds of the proposed solution and OSC would develop the remaining one third.
- Research and Development Center (R&DC): The R&D center was hired to provide support for this project.
- Contract Support: Contractor support was pivotal in the creation of the *Mission Engineering Process* document. This document laid the groundwork for standardizing the processes that the WatchKeeper system should be designed to facilitate. Contract support worked very closely with both the sponsor and the sponsor's representative on identifying the workflow of the end users of the system. This work helped to identify what would later be known as IVT, IOP, and OM. The goal here was to focus on the process and not on solutions or technology. The team felt that it would be prudent to truly understand the workflow inputs and outputs and functional areas they resided in before coming up with the technical solution.
- Operators and end users: Coast Guard command centers.

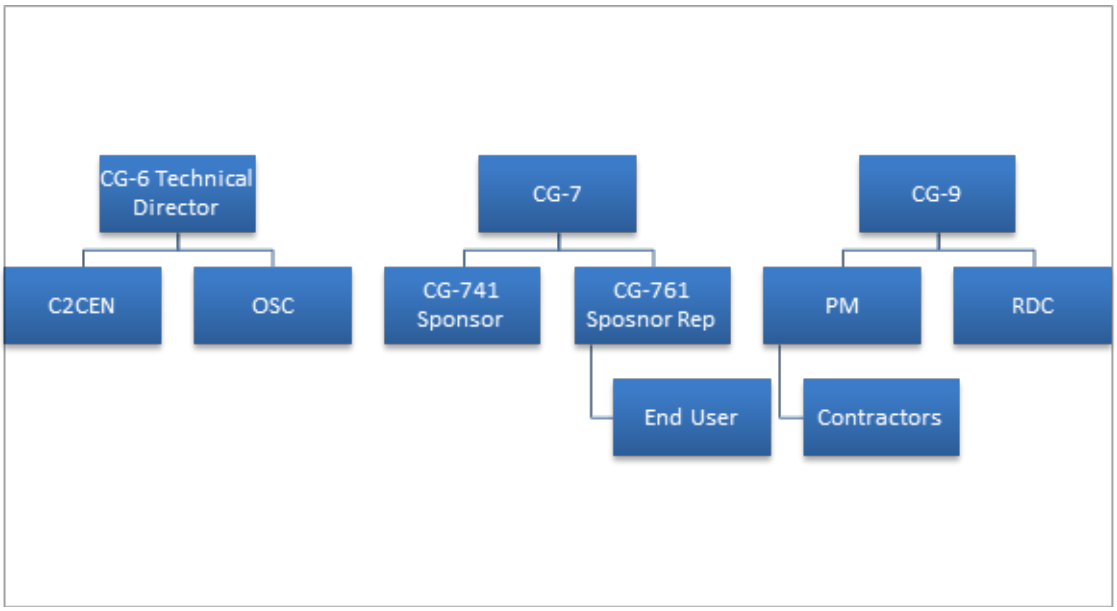

Figure 9. Stakeholder Organization

E. COMMUNICATIONS

Communication for this project was challenging given the project's size and scope. The PM established many processes to better facilitate information sharing between directorates and key stakeholders, but this was never realized. Some of the variables that challenged the PM's effort included the geographical separation of key stakeholders. C2CEN is located in Portsmouth, Virginia; OSC is the located in Kearneysville, West Virginia; CG-7 is located in the Transpoint building at Coast Guard Headquarters; and CG-9 and CG-6 are located at the Jamaal building at Coast Guard Headquarters. The challenges of being geographically separated created a logistical nightmare for the project. Stakeholders were aware of these challenges and tried to implement various forms of communications to minimize potential negative impacts. These tools included bi-weekly information management IPT meetings, weekly progress report meetings, monthly senior management team meetings, and day-to-day emails, to name a few. One of the biggest shortcomings of having physically separated key stakeholders was that too much time was wasted getting members up to speed at the beginning of every meeting. For example, despite having met two weeks ago, it took a bit of time for mid-level management to reassess and understand the issues that were last

discussed during the previous meetings. Without face-to-face communication on a regular (daily) basis, a lot of time was wasted playing catch-up. Additionally, stakeholders did not physically attend many of the meetings in person but rather attended telephonically. This only added to the inefficient and ineffective use of time.

F. OTHER FACTORS

The landscape for this project was complicated. The major stakeholder leadership was either (1) a newly promoted captain or commander or (2) a seasoned captain with many years of O6 experience. This was interesting because the new captains tried to assert themselves as seasoned captains and at times received pushback from the more veteran captains. As stated earlier, the PM was a senior O5 and was in charge of the project and, as such, had to make unpopular decisions. This had to be done very carefully because of the differences in ranks, which often led to wasted time. Given this dynamic, it was often difficult to obtain a clear picture of who was in charge of the project at any given point. On numerous occasions, CG-761 claimed that they were responsible while CG-741 felt that they were in charge. C2CEN operated as if they were in charge and held sole responsibility for deciding on the technical solution for the project.

G. WATCHKEEPER OUTCOME

1. WatchKeeper Outcome Compared to Goals and Objectives

It is difficult to classify the WatchKeeper project as a success in any capacity. The project was delivered years late, with limited capability, and was grossly over budget. There are many projects that have lofty goals, but that must settle for less because of factors outside the control of the program; this is true for WatchKeeper as well, but is not the primary reason for the project's failure. WatchKeeper's shortcomings can be attributed to many factors, all which will be explored in Chapter IV of this thesis.

H. MISSION AND ASSET SCHEDULING INTERFACE (MASI)

1. MASI Goals and Objectives

The MASI project was originally developed to support the Coast Guard IOP needs of WatchKeeper. It was the second of the three capabilities WatchKeeper was

going to deliver. MASI was going to be capable of displaying all assets, all asset statuses, and all planned, in progress, and completed missions planned. MASI was to eventually support port partner-specific planning and scheduling requirements as well with later builds. MASI was to provide a single user interface for near-real-time transparency of all asset and mission information. MASI was to support pre-planning/emergent planning, scheduling, and the execution of missions. This single presentation layer is Web based, and was to be available to anyone authorized for access to the system, including planners, watchstanders, and port partners.

Modernization places a premium on information transparency throughout the Coast Guard and DHS. This is particularly true at the Sector level, where the majority of mission execution occurs. Prioritizing missions and assigning resources are under the responsibility of the sector commander to optimize resource employment across the 11 CG-mission categories and subcategories. The effective and efficient management of resources can only occur with transparent planning and execution, and by making the results visible to all levels of command.

Mission planning is conducted via many different formats, tools, and processes. For example, spreadsheets, whiteboards, and Microsoft Outlook calendars are all used to perform planning functions within Coast Guard units. The results of this inferior process are as follows:

- The various planning products are not published in a manner that provides a single operational view to the chain of command and command centers, resulting in degraded situational awareness.
- Command centers have incomplete visibility of information on asset statuses, planned activities, assets underway, and mission completion.
- Response to emergent events is often reactionary without taking into consideration the impact of resource redeployment and without the transparency to apply risk-based decision support.
- There is low awareness of Prevention Department activities.
- After missions are complete, the missions are recorded in various enterprise authoritative databases (e.g., Abstract of Operations [AOPS], Aviation Logistics Management Information System [ALMIS], Marine Information for Safety and Law Enforcement [MISLE]) without a clear relationship between common data elements.

MASI was to provide the following capabilities and services:

- A single user interface will provide a near-real-time presentation of all resources and statuses.
- A single user interface will provide a near-real-time presentation of all mission assignments planned, underway, and completed.
- A single user interface will provide a near-real-time presentation of significant events that will influence planning decisions.
- Planners will enter planning and scheduling information and decisions in one place: MASI.
- Units and command centers will then use MASI to manage the assigned missions and to support post-mission reporting.
- A single location will be available for the display of resource and mission planning and execution, optimizing resource utilization against the highest priority missions.
- Horizontal and vertical awareness will be provided for resource and mission planning, integration, and execution.
- The requirement for reporting will not change, but the system will support standard reporting procedures.
- MDA will be enhanced by providing command centers with *single source visibility* of all activities in the area of responsibility—planned, underway, and completed.
- The system will contribute to the standardization of data management and, by extension, an increase in data integrity within authoritative systems.

To provide a better perspective on MASI's capabilities, the following is an example of the types of missions it will support:

- Resources and missions across the entire Coast Guard are displayed in one application visible to all. In the event of an emergent mission, all levels of command can see what assets are available and take the necessary actions to respond.
- By being able to observe the changing assignments and resource statuses in MASI, higher levels of command can avoid direct contact with subordinate commands and command centers, thus freeing watchstanders to better accomplish the mission.
- When a resource (e.g., cutter, boat, aircraft, or inspection team) gets underway, that movement is transparent to the command center.

- MASI captures non-asset and mission information (e.g., reasons for aborting mission, bar status, tidal closures) that is critical to operational decision-making and requirements analysis.

Figures 10–12 are screenshots of the MASI system to illustrate the previously mentioned system concepts.

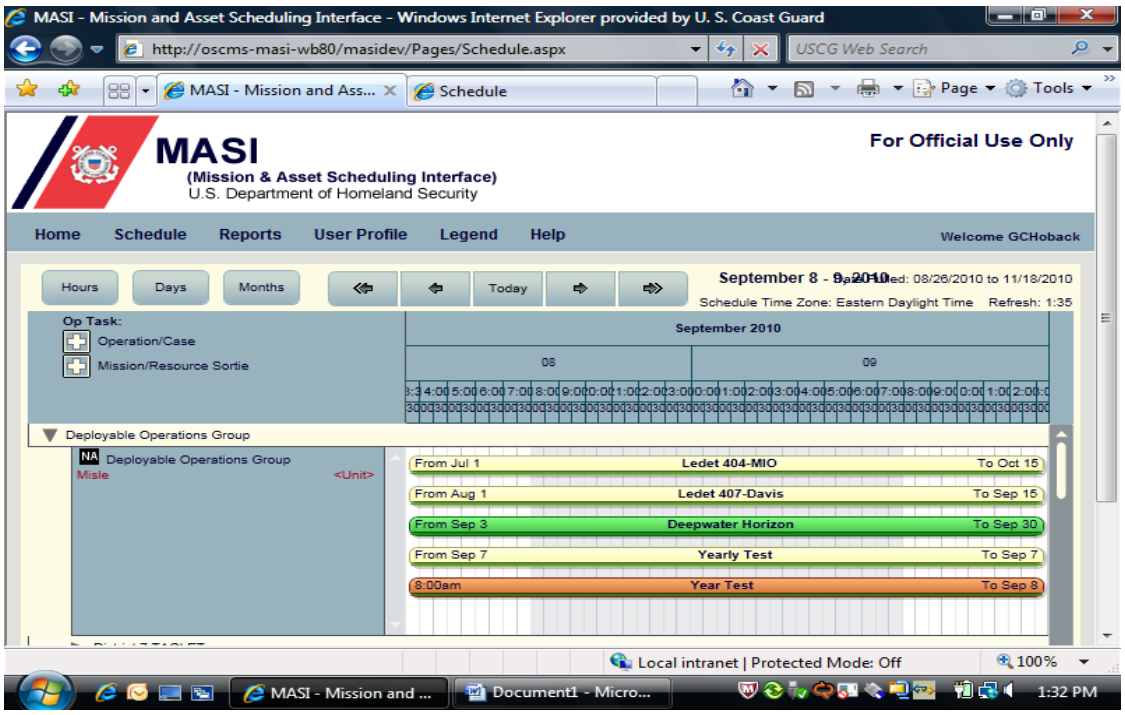

Figure 10. Overall Planning View of MASI

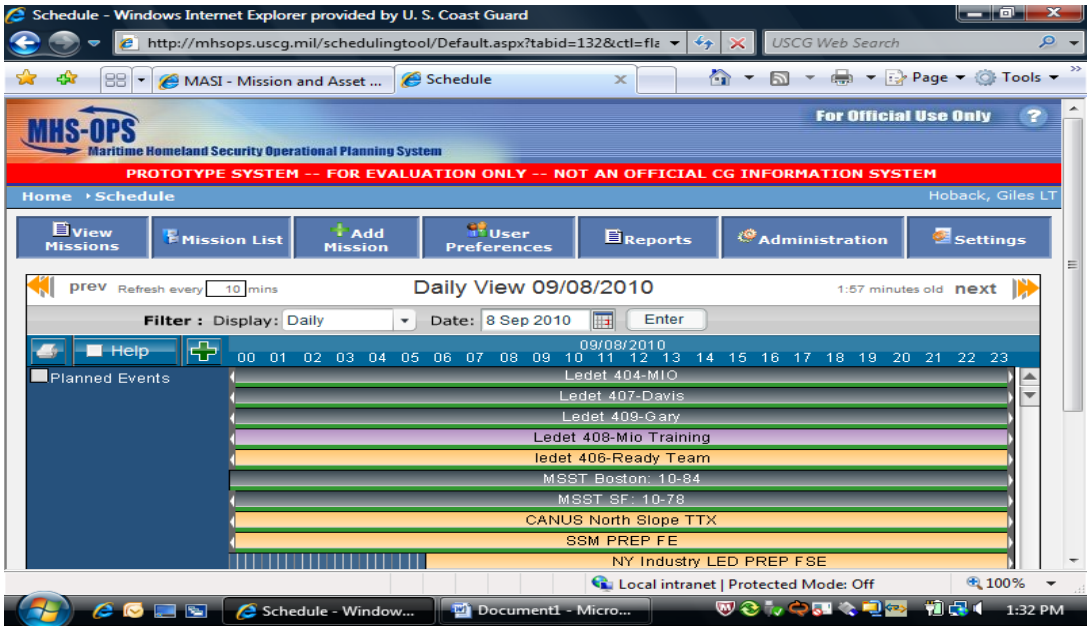

Figure 11. Overall Planning of the Prototype System Used for MASI

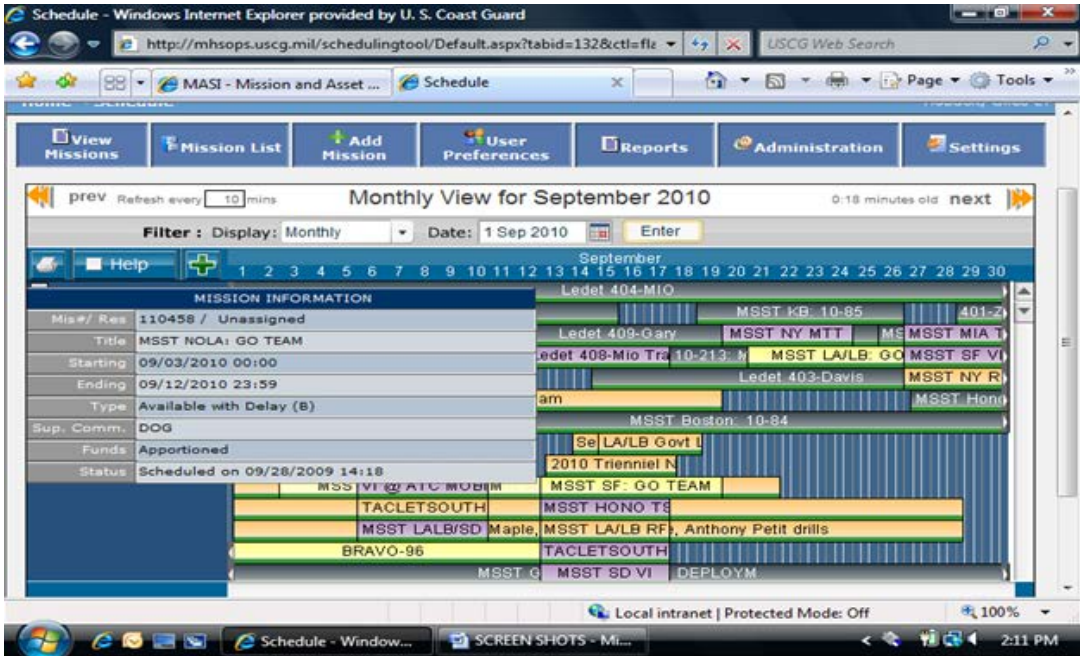

Figure 12. Fictitious Monthly View of Assets in the MASI System

I. MASI PROJECT PROCESS AND DOCTRINE

Although MASI was a component of WatchKeeper, and an IT system of its own right, it did not follow the same doctrine rigor as the WatchKeeper project. The following is the list of the doctrine that MASI used:

- *Systems Development Life Cycle* (SDLC)
- Requirements document (Excel spreadsheet)
- Testing document (Excel spreadsheet)

J. MASI PROJECT PROGRESS MEASUREMENT

The MASI project's progress was not tracked simultaneously with the WatchKeeper project, nor was MASI tracked with EVM. MASI's requirements were captured on an Excel spreadsheet with the sponsor, sponsor's representative, end user, and developers. This list of requirements was then prioritized by the end user and given to the developers to evaluate the realm of possibility and the development time needed. Once the developers completed this task, a final meeting was held and the official requirements list was generated. This list of requirements was then used to guide the development effort and track progress towards capability delivery.

K. STAKEHOLDERS, ROLES, AND RESPONSIBILITIES

- CG-741: Sponsor
- CG-761: Sponsor's representative
- CG-6: Technical agents
- OSC: Developers
- End user: Coast Guard Deployable Operational Group (DOG)
- End users: Coast Guard Command Center personnel

Figure 13 is a graphical representation of the MASI stakeholders.

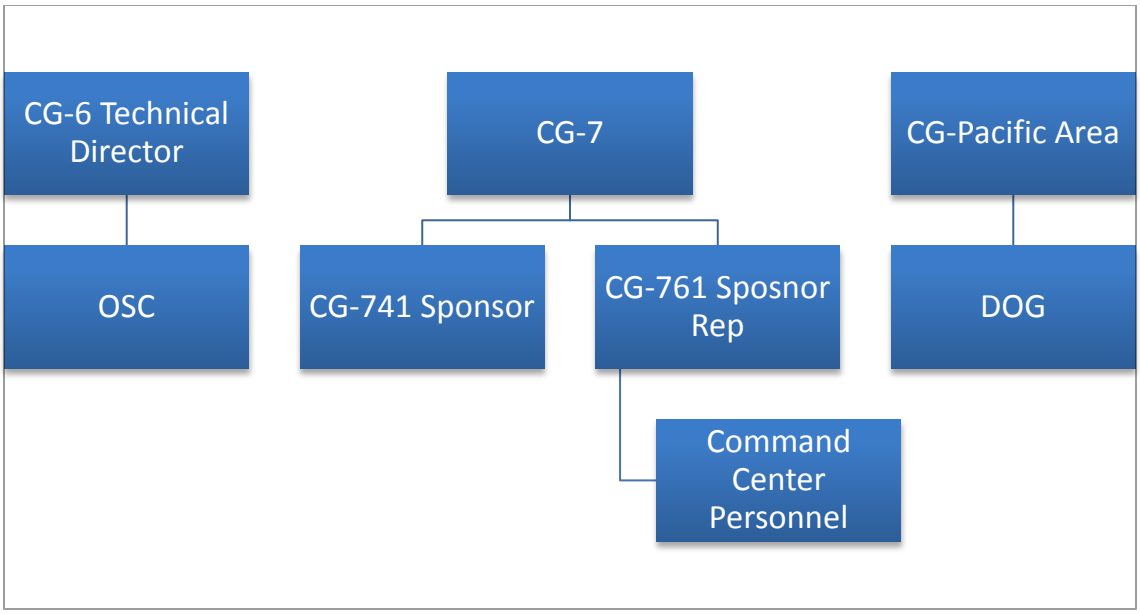

Figure 13. MASI Stakeholders

L. COMMUNICATIONS

This MASI project included daily face-to-face communication with stakeholders. The key stakeholders at Coast Guard Headquarters (CG-7, CG-6, and CG-8) were part of the daily meetings, with the support of contractors. In addition to the short, daily interactions, there was a weekly meeting that included the stakeholders at Coast Guard Headquarters, the developers from OSC, and the end users of the DOG. These meetings were face to face as well and focused on reviewing development progress and refining requirements. The luxury of meeting face to face cannot be understated. Communicating technical ideas and concepts face to face enabled the team to be more focused and committed. It nurtured a more creative, homogenous environment than other projects I have been involved with, and it facilitated storyboarding and the visual display of ideas and concepts, which was critical during the concept phase of the effort. The group size of these meetings was typically between five and seven people, and the meetings were held in a very informal setting, sometimes even standing. Ideas were mapped out on a whiteboard, and problems were worked through in a visual manner. At least one day prior to the weekly meeting, the group agreed upon a small agenda and focused only on those items. Any new business ideas or concerns were placed in a "parking lot" to be

discussed at a later time. This process proved to be especially useful in keeping the group on task and focused, given the compressed schedule of the MASI system. Another outcome of these frequent meetings was that no issues went unresolved for more than one or two days. When challenges did come up, they were discussed as a group, rather than taking a parochial perspective. This allowed somewhat of a 360° view of the issue and generally resulted in a much more fruitful solution.

Another valuable contribution to the effort was having the actual end users participate in both the daily and the weekly meetings and communicating with them face to face. Given that they were going to be the ones using the system on a daily basis, their input proved to be invaluable, and their contributions helped limit requirement ambiguities. The developers were able to explain challenges they were facing, and trade-offs could be agreed upon and understood. Additionally, priorities could be adjusted accordingly. Having access to the end user with this frequency also facilitated the development of training for the new system. Having the end user present during the development process allowed for a more thorough understanding of the system in a more contextual sense. The end user was also able to communicate actual workflow that would be required of the system, and engineering "best guesses" were eliminated from a developmental perspective.

In the MASI project, formal communication with senior leadership occurred on a weekly basis; although this communication was more frequent than communication in the WatchKeeper project, it was far less cumbersome because it lacked the WatchKeeper project's reporting requirements. The report that was generated for the MASI project was more anecdotal yet more useful when it came to relaying ideas, challenges, and the actual status of the project. The format and amount of information that was relayed during the MASI project wasn't regimented, but the content was. As a group, we felt that it was more important to capture accurate information rather than a specific amount of information. If there was nothing to report, then there was nothing to report. The group felt no obligation to fabricate information to placate leadership, and leadership appreciated this. The report that was generated represented every stakeholder's priorities, and issues that could not be agreed upon were identified as such. If issues arose from the

report, the team met within a day to address those issues. Given the political and technical issues surrounding the MASI project, this form of communication helped to ease any concerns in a timely manner. This frequent reporting also facilitated the means for any course corrections that senior leadership felt prudent to address quickly.

M. MASI: OTHER FACTORS

As previously stated, the MASI project was originally scripted to be one third of the WatchKeeper project (with the other two thirds being IVT and OM), but was directed to move out independently from the WatchKeeper effort. The reason for this push was that the designated approving authority (DAA) deemed MASI's predecessor Maritime Homeland Security Operations (MHSOPS) to be a security risk to the Coast Guard enterprise. The MHSOPS system was used as the prototype for the MASI project and provided an operational capability to the DOG. The DOG on a daily basis used MHSOPS, and if turned off, it would critically reduce the unit's operational effectiveness. Therefore, there was pressure to deploy WatchKeeper, because of the security risk that MHSOPS posed—yet at the same time, the project team had to develop the system to meet the workflow of the DOG. From the DOG's perspective, it did not want a new system because the system it was using already worked, and it did not want the aggravation of having to learn a new way of doing business. The leadership for the DOG was extremely concerned about turning MHSOPS off because it was their primary IT tool used for missions, and they had little faith that the new system (MASI) could be fielded in a timely fashion. Therefore, getting the DOG's buy-in and commitment was crucial.

From an acquisition and project management perspective, the challenge was in trying to use existing doctrine (SDLC) to guide the MASI project within the compressed timeline. The DAA, the official in charge of assessing the risk of a system within an enterprise, ordered that MHSOPS be off-line within 90 days, thereby marking the line in the sand for the delivery date for MASI. The SDLC's requirements could not be met in the timeframe established by the DAA or in the timeframe, in which MASI was being generated, so therein was the real challenge. The DAA, which is part of CG-6 and which

owns the SDLC process, was requiring MHSOPS to be turned off and MASI to be deployed in a timeframe outside the realm of possibilities with respect to meeting the SDLC mandates. This quagmire created tension within CG-6 that carried over to the MASI project team. How could the team meet both requirements? The MASI team members attempted to establish a quasi-SDLC approach, and tried to customize the documentation requirements of the SDLC to meet the MASI project needs, but this was an exercise in futility given the delivery schedule of the project. Another challenge was that the engineering approach in use to deliver the system did not dovetail with the SDLC requirements, regardless of how hard the group tried to make it fit. The team was meeting more often and delivering requirements without having required documents generated. This issue was eventually resolved with the agreement among team members and leadership that the only required documents for the initial MASI effort would be requirements documents and testing documents. The other mandated items would be addressed in future builds and when MHSOPS was off-line.

N. MASI OUTCOME

I would consider the MASI project a success, as it was delivered on time to meet the security risk identified by the DAA, while meeting the requirements of the DOG (end user). The project went from concept to delivery within three months with very little funding. With respect to MASI's contribution to the WatchKeeper project, this integration still had not happened at the time this thesis was written. However, this is not the fault of the MASI project not having the required capability, but rather the WatchKeeper project's inability to integrate the two systems. I will explore the MASI project's outcome further in Chapter IV.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. PROJECT IMPACTS

In this chapter, I provide the important variables for both the WatchKeeper and MASI projects. I analyze and interpret variables that impacted the projects, using a similar format to Chapter III. I examine the variables, process and doctrine (rigidity), progress measurement, stakeholders, communication, and other factors of both projects. I then provide a relative score on a scale of 1 to 10 of these variables on how well each respective project did in that area. For example, if the project did well in that area, the score would be closer to 10, if the project did poorly, the score would be closer to 1. I also provide the effect on cost, schedule, and performance the respective variable did. For example, I examine the *progress measurement* variable for both the WatchKeeper and MASI projects; I rate it with a relative score, and then provide the *effect* of that score on the project's cost, schedule, and performance. Finally, the chapter ends with a comparison of both the WatchKeeper and MASI projects with agile system development.

A. WATCHKEEPER PROCESS AND DOCTRINE (RIGIDITY)

As I stated in Chapter III, WatchKeeper used the MSAM and acquisition life-cycle process to manage the project. These processes are defined by upfront planning, formal documentation, and a linear phase approach, and do not provide an opportunity for the program to move back to previous phases (Benito, Casagni, Mayfield, & Northern, 2010). WatchKeeper was to deliver new capabilities that had never been delivered with previous IT systems. As such, WatchKeeper by definition was filled with many uncertainties with respect to requirements, yet because of the doctrine that was being followed required that these requirements be accurately defined upfront. One factor that led to the uncertainty with requirements was that the end user had a difficult time defining what was needed or desired. This uncertainty with requirements may be the primary reason for the difficulties the WatchKeeper project faced (Atkinson, Crawford, & Ward, 2006). Adhering to the heavyweight process outlined with the MSAM and acquisition mandates, forced the program and project to get everything right the first time, which is an impossible task if the desired results are uncertain.

Additionally, these heavyweight processes were designed for large weapon systems and not IT systems (Benito et al., 2010). As Duquette et al. (2008) describes, "Typically, the acquisition development cycle is quite lengthy, as much as 5 to 10 years; and this development cycle is too long for IT programs" (p. 25). By the time these technologies are fielded, they are outdated and no longer address the needs of the end user.

The Joint Capabilities Integration and Development System (JCIDS) is the process in the acquisition cycle that is designed to reduce uncertainty with development by focusing on needed capabilities, rather than future threats. Although the JCIDS process is an effort to reduce ambiguity with the requirements phase of the process, it is, in my opinion, still far too cumbersome for the delivery of IT systems. As such, I feel that the WatchKeeper project was handcuffed from the very beginning because of the uncertainty with the requirements and that an agile approach would have produced better results. WatchKeeper's shortcomings can be illustrated with an excerpt from a 2010 letter from the sponsor's representative (CG-761) to the PM (CG-9):

> As the Sponsor Representative for the information management (WatchKeeper) component of the Interagency Operations Center I have serious concerns with the current status, progress, and direction of the project. The project has had several system scope changes, has missed every capability delivery date, and is currently months behind schedule on the next deliverable. These delays reach beyond the project, and are beginning to negatively impact both the Sector Command and port partners alike. For example, the Operational Testing and Training schedules have been rearranged 3-times with Sector personnel, putting an unnecessary burden on an already over-taxed workforce. The root cause of these issues stems from the failure to implement basic project management tools, including an Integrated Master Schedule and an appropriate Integrated Support staff to meet the requirements and mandates of the project. I am no longer confident that our system development aligns with our operational requirements; therefore, I am again requesting detailed architecture views/diagrams, Integrated Project Plan, a summary report of the technical challenges encountered thus far and the action officer assigned to solve these challenges. (Sponsor representative, personal communication, October 4, 2009)

Clearly, there were concerns from the sponsor's representative perspective about the management approach taken to deliver the WatchKeeper project. Many of the project's managerial challenges were not due to competence issues but rather to the fact

that the PM had no alternative approaches other than the one dictated by the MSAM and the acquisition process mandates. Many of these mandates added no value to the process but had to be complied with because of the dollar threshold that WatchKeeper met. The inflexibility of these policies increased the cumbersomeness of an already complex project. These policies prevented opportunities for program management to seek alternative engineering approaches more suited for a project with these characteristics (e.g., unknown requirements, use of new technologies, large team size, and geographically separated stakeholders) and forced the use of traditional system development applications. As such, this traditional approach required that a plethora of documents be generated—many of which called for detailed information that was unknown given the uncertainty of the project requirements.

Additionally, the actual system development approach and solution were chosen prior to establishing a requirements document. There was the nonspecific pORD that outlined and identified ideas and concepts at a very high level, but by no means did it provide the fidelity needed to develop an actual system or engineering solution. A detailed requirements document did not exist because at this point, the end user (via the sponsor's representative) had not identified them. The *Mission Engineering Book*, which would later be delivered to show workflows and business processes that WatchKeeper would facilitate, had not been developed yet, and the missions that were to be accomplished were not completely identified. Regardless of these facts, C2CEN set out to start developing the system, using the pORD as guidance (Appendix 1). To add to this problem, the developers derived the original system requirements alone—without other stakeholders present. The impact of the first iteration of system requirements generated by the developers in a silo would prove to be an issue that the project never recovered from.

B. WATCHKEEPER PROJECT PROGRESS MEASUREMENT

Meeting requirements was a challenge for the WatchKeeper project. The project requirements were not identified, so it was impossible to understand 100 percent of the project's scope. As stated, EVM requires full disclosure of actual results so that all

stakeholders receive the same information and allows only one set of books. There were several attempts made to accomplish this, but there was never 100 percent buy-in from stakeholders due to a lack of trust. A measurable project baseline must be identified and locked down for EVM to hold any value, but this never happened. There were attempts to comply with locking down the EVM variables, but because there was not a clear understanding of requirements, this was an impossible objective. Finally, EVM requires that a forecast be made periodically, but because of moving baselines, shifting delivery schedules, a number of requirements to be fulfilled per release, and closed communication among stakeholders, the EVM effort proved to be a waste of time for WatchKeeper.

Although a valid attempt was made to establish EVM requirements as mandated by the MSAM doctrine, at no time was WatchKeeper close to meeting and achieving a 90 percent success rate on cost, performance, and schedule goals, as identified by the DHS (2009) requirement. The lack of accurate EVM data contributed to the WatchKeeper project's failing to meet acquisition milestones on time, resulting in a loss of faith at the DHS level. The WatchKeeper project failures with EVM are not a reflection of EVM techniques and their usefulness; it is a reflection of poor EVM execution. If the requirements for WatchKeeper had been understood, if communication channels had been forged, or if the project had not been handcuffed with cumbersome mandates from the acquisition process, the likelihood of EVM success would have been far greater. The WatchKeeper project's failure to implement EVM does not suggest that EVM was incorrect; it suggests that the Coast Guard could not properly facilitate the requirements of EVM.

C. STAKEHOLDERS AND COMMUNICATION

Additional factors that led to challenges of the WatchKeeper project included a lack of cohesiveness among key stakeholders, a failure to communicate effectively, a lack of trust, the geographic separation of stakeholders, and a sense that there would be endless funding for the project. It did not matter what was done; it could always be fixed. On more than one occasion, various stakeholders—including at the leadership level—

mentioned that the first version of WatchKeeper did not count and that it was disposable. This philosophy removed any feeling of accountability from stakeholders and perpetuated an environment that fostered complacency and stakeholder independence. Delivering useful code did not appear to be the primary focus; instead, meeting the mandates of the acquisition process and the MSAM was the priority, since failure to do this would surely derail the project at the DHS level, which would lead to a loss of funding. As a result, stakeholders worked diligently, but independently, on delivering their required portion of the MSAM documents, with little care as to the actual requirements or system needs.

As mentioned in Chapter III, WatchKeeper communication efforts often were in vain. Many of the meetings were held at a distance because of the geographical separation of the stakeholders. These consequences of non-face-to-face meetings were misconceptions of information passed between stakeholders. Each directorate had its in-house method for internal communication and its own dialect, but when these various lexicons were brought together in a group setting, they did not necessarily result in a clear understanding of the message. Document control and management were also challenges. Despite having a consensus that it was important to maintain document control, leaders of the project never could properly manage documents. There were many meetings where two different versions of a document were being reviewed simultaneously, and a lot of time was wasted simply trying to identify the appropriate artifact to discuss.

D. WATCHKEEPER OTHER FACTORS

The WatchKeeper stakeholder dynamics were interesting. Aside from the organizational and hierarchy challenges, the biggest hurdle was the political landscape that existed amongst stakeholders. This politically charged jockeying ended up being a true detriment to the project. Besides the normal disagreements and uncertainties that are present in any project, this project had a level of animosity between stakeholders because of military ranks that were involved. There were meetings where quarreling dominated the agenda, and there was a lack of trust between stakeholders that at times bordered on resentment. C2CEN felt that nobody trusted its efforts, while both directorates in CG-7 felt that C2CEN was not being honest with the development efforts that were underway.

CG-6 had an interesting role: C2CEN is typically tasked by CG-6, but because this project was a major acquisition, CG-9 was in charge and directed C2CEN, which presented internal challenges within both CG-9 and CG-6.

Senior leadership also introduced pressure to the WatchKeeper. It was often said by senior management that this project "was too big to fail." Therefore, the information that was passed to the decision-makers was often a more positive perspective than reality. No group was willing to be responsible for the failure of the project. Milestone deliverables and expectations were all managed in a way that would present the organizing group in the best light. From a program management perspective, it was very difficult to gauge the true pulse of the project given these realities.

Another challenge for the WatchKeeper project was that the developers, based on their interpretation of the pORD document, derived the WatchKeeper system requirements independently. This introduced many challenges to the delivery of the WatchKeeper system. The developers decided which requirements to deliver and when to deliver them. Initially, the developers broke the requirements into three spiral deliverables. The first spiral would deliver eight percent of the requirements, the second spiral was slotted to deliver 12 percent of the requirements, and the third spiral would deliver the remaining 80 percent of the requirements. After missing the delivery date of the first spiral by 114 days, the developers reduced the targeted scope by 50 percent and added five additional spiral releases. Again, these decisions were made independently without input from other stakeholders.

The WatchKeeper project also failed to meet testing events. Because of this failure, the Coast Guard finally decided—with pressure from the DHS—to reduce the scope of WatchKeeper. Therefore, in 2010, the DHS gave the direction that WatchKeeper was to be deployed as a technology demonstrator rather than a full-fledged system of record, which removed the MSAM requirements from the WatchKeeper effort. This decision came at a price. The WatchKeeper project realized substantial funding cuts, and there was operational backlash as well. At the time of writing this thesis, WatchKeeper is still being deployed throughout the nation at Coast Guard SCCs as a technology demonstration, with far fewer capabilities than envisioned.

E. MASI OUTCOME COMPARED TO GOALS AND OBJECTIVES

The MASI project was delivered on time, but more importantly, it met the operational needs of the DOG. Therefore, MHSOPS could be taken off-line and the security risk to the enterprise was removed. Another success of MASI was the effectiveness of the training that was established and the execution of this training to the DOG in their subordinate units. The success of this training is a direct result of having the DOG representation during the development process. Not every desired capability was delivered with the first iteration of MASI, but the system that was delivered could be used effectively to accomplish the DOG's missions. Without doubt, there is a direct correlation between the success of the project and the development approach taken to deliver the system.

F. MASI PROCESS AND DOCTRINE

Another factor, and quite possibly the most important one with respect to MASI's success, was the reality that the project did not get bogged down with documentation and paperwork. Although it did follow the SDLC as outlined in Chapter III, it was a modified SDLC that only required documentation that was beneficial to the development efforts. Having flexibility within this process was incredibly useful to stakeholders. There was no expectation to simply "check the box" for paperwork drills. It was not as if the MASI project did not follow a process or create documentation; the MASI project simply was allowed to modify established procedures to facilitate a more useful development approach. Time was of the essence given the security risk identified by MHS OPS, and the flexibility allowed for tailoring the regimented process was significant.

G. MASI PROGRESS MEASUREMENT

Another contribution to the success of MASI was the manner in which progress was assessed. The metric that was used for assessing the progress of the project was not EVM like that of WatchKeeper, but rather actual capability delivered by the developers. As stated earlier, priorities were established during the daily face-to-face meetings, and the developers used these priorities as a recipe for delivering the system. During the weekly meetings, progress reports on these priorities were presented to the group, and on

a bi-monthly basis, tangible system capabilities were demonstrated. Another crucial component of the delivery of the MASI system was the management of expectations. By no means was the first release of the system expected to be the end-all and be-all, but rather it was viewed as a first foundational step in a series of releases, and everybody was aware of this. Again, the frequent meetings, the establishment of an agreed-upon direction, access to a prototype system (MHSOPS) as a guide, and honest stakeholder communication simplified many of the complexities that typically hinder progress in system development efforts.

H. MASI STAKEHOLDERS AND COMMUNICATION

Face-to-face daily communication was also critical in the delivery of MASI. I cannot remember an instance where a key stakeholder was unaware of the progress of the project or the immediate future goals of the effort. Face-to-face interaction was the driving force behind this. In addition, having the developer, the customer, the testing team, and the enterprise team communicate in the manner in which they did knocked out many obstacles and ambiguities typical of a software development effort. As outlined in Chapter II, people are the driving force behind successful software development deliveries. This was realized with the MASI project.

I. MASI: OTHER FACTORS

Although the MASI project was successful, and a capability was delivered to the operator, the MASI project still has challenges ahead. As with any endeavor, momentum must be maintained, which requires that leadership continue to support the effort. There is still the need to integrate MASI into WatchKeeper, and this is going to present some challenges to both the WatchKeeper and MASI projects. With that said, if the appropriate level of importance and support is given, there is no doubt that the MASI and WatchKeeper integration effort will be successful.

J. WATCHKEEPER AND MASI PROJECTS RELATIVE SCORE

As described at the beginning of this chapter, I will provide a description and metric value for variables that impacted both the WatchKeepr and MASI projects. The

variable will be underlined, the project in reference bolded, and the relative score follows the related project. The impact of the variable is then explained in relation to the effect it had on the project's schedule, cost, and performance, which is identified in italics.

Progress measurement: the importance of measuring the progress a software development effort cannot be undervalued. A baseline must be established and locked down so that a road map can be established. Progress measurement is the metric that is used to communicate work that has been done and work that remains, to both stakeholders within the project and interested parties outside the project.

WatchKeeper

How well the WatchKeeper project did with *progress measurement* on a scale of 1-10 (1= low, 10= high): 2

Effect

Schedule: although EVM was used for the project, the deliverables were never base-lined or locked down. Requirements were moved from date to date, or deleted all together by the developers. As such, clear deliverables were never established; what was being delivered and when was never clear, which made scheduling extremely difficult. Milestones were established and missed. When this happened, another schedule was established and new milestones were identified. These new milestones contained more deliverables than the previous milestone, and typically less time.

Costs: the impact from the lack of an honest progress measurement tool is obvious. The WachKeeper project could not definitively express progress within the project because of the lack of implementing a progress measurement tool. This had a negative impact on the costs of the project.

Performance: with the requirement delivery schedule never being established, it was unclear as to what the final capabilities of the system would yield. To date, the WatchKeeper system still has not successfully passed Key Performance Parameters (KPP's) and Critical Operating Issues (COI's) tests, and is still being fielded as a technology demonstration to end users.

MASI

How well the MASI project did with *progress measurement* on a scale of 1-10 (1= low, 10= high): 7

Effect

Schedule: MASI was a smaller project in scope and therefore easier to manage with respect to schedule. There was a prototype to work from (MSHOPS) and the number of requirements identified for delivery was minuscule compared to that of WatchKeeper. As such, the scheduling was realistic with end user priorities being the focus of delivery. If the requirements were not a priority and an engineering possibility given time or technical skill, the requirement was pushed to a later iteration/deliverable. These issues were identified during the daily face-to-face meetings with stakeholders.
Cost: the majority of the capabilities were delivered to the end-user within budget.

Performance: the majority of the capabilities was delivered to the end-user and met the identified requirements.

Stakeholder: I have broken down stakeholders into the following categories: trust among stakeholders, stakeholder professional experience, stakeholder proximity to each other geographically, stakeholder support of the project, and finally stakeholder turnover. Trust amongst stakeholders is vital for successful software development efforts. There must be a genuine trust of each other so that key metrics of progress have validity. Professional experience with the stakeholder's respective role is important to understand within a project so that adequate time can be allotted for training as necessary, and expectations can be managed with expected time to complete a task. Obviously, the more experience a stakeholder has in their respective role, the less time would be needed for training and theoretically the more experience a stakeholder has, the quicker a task can be completed. Stakeholder geographic proximity with each other affects the manner in which the stakeholders communicate, perform their respective tasks, and interact with each other. The closer the stakeholders are, the easier it is to perform these functions. Stakeholder Support of the project is critical. Without genuine support of all stakeholders

involved with the software development project, the project will be hampered. By support, I am not only referring to funding, but I am also referring to staffing, leadership support, belief in the project, and how the project will be integrated within into the overall enterprise of the organization. Finally, stakeholder (personnel) turnover reflects the impact of project stakeholders leaving a project and being replaced. Stakeholder turnover happens for a myriad of reasons, and is especially prevalent within DoD and DHS given rotations, promotions, and changing priorities. As such, the impact of this disruption must be realized, and the impact on the project's progress must be understood. When new stakeholders are brought on to a project, this individual must be trained in the technology, the goals, and the overall strategy of the project. This adds time to the project.

WatchKeeper

How well the watchKeeper project did with the *stakeholder* variable on a scale of 1 to 10 (1= low, 10= high):

- Trust among stakeholders: 4
- Stakeholder professional experience: 4
- Stakeholder proximity to each other: 3
- Stakeholder support of the project: 7
- Stakeholder turnover: 3 (many stakeholders left the project)

Effect

Schedule: the lack of *trust* among stakeholders had a negative effect on the schedule. Stakeholders often withheld information from each other, including progress information and information about delays. The *professional experience* of stakeholders was fairly low. Both the sponsor and sponsors representative had no prior experience in their role prior to the WatchKeeper project. Developing the WatchKeeper system was also a first for the developers of the project. Although they had prior experience with system development, never had they taken a project on of this magnitude. The program manager of the project had experience with acquisition and program management, but never had the program manager been responsible for a project of this size. The inexperience of stakeholders had a negative effect on the schedule. Stakeholders were

geographically separated as mentioned earlier this thesis, and as such added a layer of complexity to many facets of the project. This geographical separation also had a negative effect on the schedule. *Stakeholder support* of the project was relatively high. All of the stakeholders wanted the project to succeed, and all of the stakeholders realized the value of the project for the Coast Guard. Stakeholder support did not have a negative effect on the schedule. *Stakeholder turnover* for the project was high. As mentioned earlier in the thesis, roles and responsibilities were interchanged and stakeholders left the project for various reasons. As such, the new stakeholders coming into the project required time to come up to speed with the happenings of the effort. This had a negative effect on the project schedule.

Cost: the above-mentioned stakeholder variables had a negative impact on the project, resulting in the cost being driven higher.

Performance: the above-mentioned stakeholder variables had a negative impact on the project, resulting in reduced capabilities being delivered to the end-user.

MASI

How well the MASI project did with the *stakeholder* variable on a scale of 1 to 10 (1= low, 10= high):

- Trust among stakeholders: 9
- Stakeholder professional experience: 5
- Stakeholder proximity to each other: 7
- Stakeholder support of the project: 9
- Stakeholder turnover: 10 (no stakeholders left the project)

Effect

Schedule: of the above-mentioned stakeholder variables, the only variable that impacted the schedule in a negative way was stakeholder professional experience. The developers of the project had experience working with the technology being implemented, and the program manager of the effort was comfortable managing MASI project. The sponsor and sponsors representative were again relatively new to their respective roles, but given their experience gained from the WatchKeeper project, and the

high experience of the other stakeholders involved in the effort, the impact was minimal to the schedule.

Cost: the MASI project was delivered within budget.

Performance: the MASI project delivered the capabilities identified in the requirements.

Communication: there are several forms of communication that I'm referring to. They are *Formal* (meetings, testing events, requirement generation), *informal* (elevator, water-cooler, lunches, etc), *written (*both official project documents and adhoc email for example), and *team size.* Regardless of the form, the communication must be open and available to all stakeholders. The final aspect of communication is *team size*. As pointed out by Brooks, Jr., (1982, p. 18), "communication is made up of two parts, training and intercommunication. Of the two, intercommunication is worse. As tasks are separately coordinated, the effort of intercommunication increases *n(n-1)/2*. For example, three workers require three times as much pairwise intercommunication as two; four requires six times as much as two, etc." Therefore, the more stakeholders that are involved, the more complex the communication variable becomes.

WatchKeeper

How well the WatchKeeper project did with the *communication* variable on a scale of 1-10 (1= low, 10= high):

- Formal: 5
- Informal: 7
- Written: 3
- Team size: 4

Effect

Schedule: of all of the communication variables mentioned above, the biggest detriment to the WatchKeeper project was with the *written* and *team size* variables. Given that the trust among stakeholders could have been stronger, there were many written communications that only reached certain stakeholders, and were purposefully withheld from others. This includes actual project memorandums that were not routed to certain

stakeholders for various reasons. Additionally, the team *size* of the project facilitated the gravitation into "cliques" amongst stakeholders. These "cliques" shared emails and other items within their group, but not outside. Many of these communications related to the schedule, and because they weren't open and available to all, the schedule was impacted.

Cost: a lot of time was wasted given the lack of strong communication channels, and as such, the cost of the project was impacted negatively.

Performance: again, given the lack of communication amongst stakeholders had a negative effect on the performance and functionality of the WatchKeeper project.

MASI

How well the MASI project did with the *communication* variable on a scale of 1 to 10 (1= low, 10= high):

- Formal: 7
- Informal: 9
- Written: 9
- Team size: 9

Effect

Schedule: there was no negative impact on schedule because of communication within the MASI project.

Cost: the MASI project was not negatively impacted because of the communication variables.

Performance: given the frequent face-to-face meetings, and the small team size, any risks to the project were dealt with immediately and understood by all stakeholders. As such, communication had a positive impact on the delivery of capabilities to the end-users.

Rigidity of the development process being followed: the appropriate development process that should be selected is dependent on the project needs. For new technologies and uncharted efforts, flexibility is paramount. Obviously new technology requires more

time to understand, forecast, develop, and implement. However, for more routine efforts, or for maintenance and support, standardized processes might be appropriate.

WatchKeeper

How well the WatchKeeper project with the *rigidity* variable on a scale of 1-10 (1= low, 10= high): 3

Effect

Schedule: because of the required mandates of the MSAM and other acquisition policies, the WatchKeeper project was handcuffed in such a manner that tasks were being assigned just to "check the box", despite little to no value being added to the overall success of the project. Many man/woman hours were wasted "checking the box", and because stakeholders realized that these activities had no impact on the project, the motivation to complete these tasks were extremely low. Additionally, the technology was new to the developers of WatchKeeper, so there were many times that they were learning "on the fly". However, the mandate of the acquisition process requires that project needs and engineering solution be identified upfront with little to no time for updating. Given the new technology and experience of the developers, it was almost impossible to clearly identify when capabilities would be delivered despite best efforts. These activities had a negative effect on the schedule of the project.

Cost: the above-mentioned factors negatively impacted the costs of the project.

Performance: given the technology challenges, many of the capabilities were not delivered as identified in the requirements.

MASI

How well the MASI project did with the *rigidity* variable on a scale of 1 to 10 (1= low, 10= high): 9

Effect

Schedule: the MASI project was not riddled with having to meet specific mandates. Although the SDLC was the guidance used for the effort, the project was given flexibility as to which sections with in the SDLC would be followed. If the stakeholders

felt that a certain function would add no value to the effort, it was skipped. This flexibility enabled stakeholders to focus on value added processes, and as such the schedule of the project was met.

Costs: cost was not impacted with the process implemented to develop MASI.

Performance: all capabilities were delivered as identified in the requirements, and the flexibility afforded to the stakeholders was a critical reason for this.

Other factors: outside pressures refers to situations such as, political, time to develop the project ("this project has to be done by this date, no exceptions), etc. The more of these variables that are introduced to the project, the more likely shortcuts are going to try to be taken. Shortcuts do not have a positive effect within the development effort, at worst, they lengthen the effort.

WatchKeeper

How well the WatchKeeper project did on a scale of 1-10 (1= low, 10= high): 5

The WatchKeeper project had outside factors that impacted the project. For example, given the hierarchy of the stakeholders, there were political influences of the effort. Another critical factor that impacted the project was that the WatchKeeper project was classified as "too big to fail", and as such, the true reality of progress was never ascertained or accepted.

Effect

Schedule: the "other" factors had a negative impact on the schedule of the project.

Costs: the "other" factors had a negative impact on the schedule of the project.

Performance: the "other" factors influenced the delivery of capabilities for the project and as such had a negative impact on the effort.

MASI

How well the MASI project did with the *other factor* variable on a scale of 1 to 10 (1= low, 10= high): 7

The only "other" factor to impact the MASI project was the pressure to deliver something quickly so that MHSOPS could be taken off-line. This pressure was both negative and positive to the project effort. The negative aspect was the identified compressed timeline given to the project by leadership. The positive aspect of the "other" factors was that leadership was motivated to get MHSOPS off-line given the security risk it posed to the enterprise, and as such provided timely support as needed.

Effect

Schedule: the "other" factors had a positive impact on the schedule of the project given the reasons identified above.

Costs: the "other" factors had a positive influence on the project as identified above.

Performance: the "other" factors had a positive influence on the project as identified above.

Table 6. Aggregated totals of WatchKeeper and MASI relative scoring

	WatchKeeper	**MASI**
.Progress Measurement	76. 2	77. 7
.Stakeholder:	84.	90.
. Trust	85. 4	91. 9
. Experience	86. 4	92. 5
. Proximity	87. 3	93. 7
. Support	88. 7	94. 9
.Turnover	89. 3	95. 10
.Communication	101.	106.
. Formal	102. 5	107. 7

	WatchKeeper		MASI	
. Informal	103.	7	108.	9
. Written	104.	3	109.	9
0. Team Size	105.	4	110.	9
1. Rigidity	112.	3	113.	9
4. Other Factors	115.	5	116.	9

K. WATCHKEEPER AND MASI PROJECTS COMPARED TO AGILE DEVELOPMENT

Table 6 shows both the MASI and WatchKeeper projects compared to Boehm and Turner's (2004) theory of the five critical factors involved in determining the relative suitability of agile or plan-driven methods given a project situation. The blue-shaded boxes reflect the plan-driven approach, while the green-shaded boxes reflect suitability more appropriate for agile methods. As clearly identified by the table, the MASI project was better suited for an agile approach given the factors involved with the project, while the WatchKeeper project was almost split between plan-driven and agile given the factors involved.

Table 7. The Five Critical Agility/Plan-Driven Factors: Comparison With WatchKeeper and MASI Projects (From Cockburn et al., 2005, p. 55)

	WatchKeeper	**MASI**	**Agile**	**Plan Driven**
Size (Number of personnel on the team)	Upwards of 20 people	6–8 people	Well matched to small products and teams. Reliance on tacit knowledge limits scalability.	Methods evolved to handle large products and teams. Hard to tailor down to small projects.
Criticality (The impact of software defects in terms of comfort, money, and or lives)	Low	Medium but closer to low	Untested on safety-critical products. Potential difficulties with simple design and lack of documentation.	Methods evolved to handle highly critical products. Hard to tailor down to low criticality products.
Dynamism (The degree of requirements and technology change)	Ambiguous changing requirements	Changing technology	Simple design and continuous re-factoring are excellent for highly dynamic environments but a source of potentially expensive rework for highly stable environments.	Detailed plans and big design up-front are excellent for a highly stable environment, but a source of expensive rework for highly dynamic environments.
Personnel (Technical abilities as defined in Table 3)	Low for the task at hand	High	Requires continuous presence of critical mass of scarce Level 2 or 3 experts (as defined earlier). Risky to use non-agile Level 1B people.	Need for critical mass of scarce Level 2 and 3 experts (defined earlier) during project definition but can work with fewer late in the project. Can usually accommodate some Level 1B people.
Culture (Whether the individuals on the team prefer predictability or can tolerate change)	The team was not designed to be flexible, nor did the team feel empowered.	Team felt empowered .	Thrives in a culture where people feel comfortable and powered by having many degrees of freedom (thriving on chaos).	Thrives in a culture where people feel comfortable and empowered by having their roles defined by clear policies and procedures (thriving on order).

An interesting observation regarding the data in Table 6 is that the data reflected in the cells of the WatchKeeper and MASI projects are the actual values from each respective project for the corresponding variable in the row. For example, the *personnel*

factor for the MASI project was high given the technical abilities of the project team, while for the WatchKeeper project, the *personnel factor* was low, given the abilities of WatchKeeper's project team. Given the technical challenges of the WatchKeeper project outlined earlier, this factor should have rated *high* as well, and as such would have switched that cell's value to green suggesting that an agile approach would have been more appropriate. The *personnel* factor was not the only factor that was in error for the WatchKeeper project. The *culture* factor cell, which is a gauge of how many degrees of freedom the stakeholders have, should have been green as well for the WatchKeeper project, given the uncertainty of many of the variables, such as vague requirements and new technologies exploration. As outlined in Chapter II, these types of variables are better suited for an agile approach, and as such, this box too would have been green, again suggesting that the WatchKeeper project should have used an agile approach, instead of a plan-driven approach.

V. CONCLUSION

The primary objective of this thesis was to explore and understand factors that may have contributed to Coast Guard IT projects that have delivered late and or out of scope, by exploring and comparing two IT projects; WatchKeeper and MASI. Agile software development was also examined, and a history and definitions of the various methodologies were explained and outlined. Finally, case studies for both the WatchKeeper and MASI projects were outlined and the variables that led to the success or failure of the projects were explored. The variables of the WatchKeeper and MASI projects were compared to agile system development, and an analysis was conducted to evaluate whether agile methodologies were suitable for IT projects of this kind. Given this analysis, I believe that agile methodologies are quite suitable for IT projects within the DoD and DHS, and that agile development should be another tool that should be explored as an option when developing IT systems within the government. I do not believe that agile system development is a silver bullet that will solve all software development challenges; it does, however, offer a refreshing approach to software development within the DoD and DHS.

A. FUTURE RESEARCH

This thesis focused primarily on the case studies of the WatchKeeper and MASI projects, and the variables involved with those two projects. The thesis also focused on agile software development, the characteristics of the agile methodologies, and the strengths and weaknesses of the methodologies. Further research is required on the implementation of agile system development and how it can dovetail with the DoD acquisition process and other DoD acquisition mandates.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Argyris, C., & Schön, D. (1996). *Organizational learning II: Theory, method and practice.* Reading, MA: Addison-Wesley.

Atkinson, R., Crawford, L., & Ward, S. (2006). Fundamental uncertainties in projects and the scope of project management. *International Journal of Project Management, 24*(8), 687–698. doi: 10.1016/j.ijproman.2006.09.011.

Balasubramaniam, R., & Lan, C. (2007). Agile software development: Ad hoc practices or sound principles? *IT Professional, 8*(2), 41–47.

Beck, K. (2000). *Extreme programming explained*. Boston, MA: Addison-Wesley.

Beck, K. (2003). *Test driven development: By example.* Boston, MA: Addison-Wesley

Beck, K. (2005). *Extreme programming explained: Embrace change* (2nd ed.). Boston, MA: Addison-Wesley.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., & Sutherland, J. (2001). *Manifesto for agile software development*. Retrieved from http://agilemanifesto.org/

Beck, K., & Fowler, M. (2001). *Planning extreme programming applied.* Boston, MA: Addison-Wesley.

Benito, R., Casagni, M., Mayfield, K., Northern, C. (2010). *Initiatives to the warfighter* (technical report WN080041.). Bedford, MA: The MITRE Corporation.

Benito, R., Casagni, M., Mayfield, K., & Northern, C. (2011). *Handbook for implementing agile in Department of Defense information technology acquisition* (technical report MTR 100489). Bedford, MA. The MITRE Corporation.

Boehm, B. (2002). Get ready for agile methods, with care. *IEEE Computer, 35*(1), 64–69.

Boehm, B., & Turner, R. (2004). *Balancing agility and discipline.* Boston, MA: Addison-Wesley.

Boehm, B. (2006). *A view of 20th and 21st century software engineering.* In *Proceedings of the 28th International Conference on Software Engineering* (pp. 12–29). New York, NY: ACM. doi: 10.1145/1134285.1134288

Bohner, S., & Coram, M. (2005). The impact of agile methods on software project management. In *Proceedings of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems* (pp. 363–370). Los Alamitos, CA: IEEE Computer Society.

Brooks, Jr, F.P. (1982). *The mythical man-month: Essays on Software Engineering*. Boston, MA: Addison-Wesley.

Burd, S. D., Jackson, R. B., & Satzinger, J. W. (2012). *Systems analysis and design in a changing world* [Course technology]. Belmont, CA: Cengage Learning.

Clifton, M., & Dunlop, J. (2003, September 29). What is DSDM? Retrieved from http://www.codeproject.com/Articles/5097.

Cockburn, A. (2002). *Agile software development*. Boston, MA: Addison-Wesley.

Cockburn, A. (2006). *Agile software development: The cooperative game* (2nd ed.). Boston, MA: Addison-Wesley.

Cockburn, A., & Highsmith, J. (2001a, September). Agile software development: The business of innovation. *Computer, 34*(9), 120–129.

Cockburn, A., & Highsmith, J. (2001b, November). Agile software development: The people factor. *Computer, 34*(11), 131–133.

Department of Homeland Security. (2009, October). *Department of Homeland Security acquisition manual*. Retrieved from http://www.dhs.gov/xlibrary/assets/opnbiz/cpo_hsam.pdf

Dingsøyr, T., & Dybå, T. (2008, August). Empirical studies of agile software development: A systematic review. *Information and Software Technology, 50*(9–10), 833–859.

Dingsøyr, T., Dybå, T., Moe, N. B., & SpringerLink. (2010). *Agile software development: Current research and future directions*. Berlin, Germany: Springer.

Duquette, J., Bloom, M., & Crawford, L. (2008). *Transitioning Agile/Rapid Acquisition*. In *Handbook for implementing Agile in Department of Defense*. Bedford, MA. The MITRE Corporation.

Erickson, J., Lyytinen, K., & Siau, K. (2005). Agile modeling, agile software development, and extreme programming: The state of research. *Journal of Database Management, 16*(4), 88–100.

Extreme programming. (2000). Retrieved from http://www.extremeprogramming.org/map/project.html

Feature-driven development. (n.d.). Retrieved from http://www.step-10.com/SoftwareProcess/FeatureDrivenDevelopment/index.html

Fleming, Q. W., & Koppelman, J. M. (2009, March). The two most useful earned value metrics: The CPI and the TCPI. *Cost Engineering, 51*(3), 16–18.

Glaiel, F., Moulton, A., & Madnick, S. (2013). *Agile project dynamics: A system dynamics investigation of agile software development methods* (Working Paper CISL# 2013=05). Cambridge, MA: MIT.

Government Accountability Office. (2012). *Portfolio management approach needed to improve major acquisition outcomes*. Retrieved from http://www.uscg.mil/history/docs/GAO/GAO2012PortfolioMgmt648636.pdf

Highsmith, J. (1997). Messy, exciting, and anxiety-ridden: Adaptive software development. *American Programmer*, *10*(1).

Kraut, R. E., & Streeter, L. A. (1995). Coordination in software development. *Communications of the ACM*, *38*(3), 69–81.

Lynch, C. (2011, November 4). What scrum and how do we use it? [blog post]. Retrieved from http://www.realmdigital.co.za/post/whats-scrum-and-how-do-we-use-it//

Nerur, S., Mahapatra, R., & Mangalara, G. (2005, May). Challenges of migrating to agile methodologies. *Communications of the ACM*, *48*(5), 72–78.

Pisano, J., Teece, A., & Teece, D. J. (1997). Dynamic capabilities and strategic management. *Strategic Management*, *18*(7), 509–533.

Primavera Systems. (2008, November). Leveraging earned value management and IT governance. *Contract management*, *48*(11), 66–70.

Pruitt, J. (2011, February 12). Perspectives on software development [blog post]. Retrieved from http://blog.jgpruitt.com/2011/02/12/crystal/

Scio. (2010, February 24). Lean product software development in 4 phases [Web blog post]. Retrieved from http://blog.sciodev.com/2010/02/24/lean-software-product-development-in-4-phases/

Security and Accountability for Every Port (SAFE Port) Act of 2006, 109th USC 4954 (2006).

Senge, P. M. (1990). *The fifth discipline: The art and practice of the learning organization*. New York, NY: Doubleday/Currency.

Sengupta, K. Van Oorschot, K .E., & Van Wassenhove, L. N., (2013). *Dynamics of Agile Software Development.*

Strigel, W. (2001). Using extreme programming and other experiences. *IEEE Software, 18*(6), 17–18.

Suganya, G., & Mary, S. A. (2010). *Progression towards agility: A comprehensive survey.* Paper presented at the Second International Conference on Computing, Communication, and Networking Technologies, Karur, India.

Turk, D., France, R., & Rumpe, B. (2005). Assumptions underlying agile software-development processes. *Journal of Database Management, 16*(4), 62–87. Retrieved from http://search.proquest.com/docview/199601533?accountid=12702

Van de Ven, A. H., Delbecq, A. L., & Koenig, R. (1976). Determinants of coordination modes within organizations. *American Sociological Review, 41*(2), 322–338.

Wells, D. (2011). Extreme programing project. Retrieved from http://www.extremeprogramming.org/map/project.html.

APPENDIX

DRAFT

Preliminary Operational Requirements for WatchKeeper Segment One

Capabilities and Functions Overview: The capabilities and functions below are derived from the PORO to guide initial development of Segment One in preparation for a modified mission engineering process. They are subject to revision upon completion of a comprehensive domain analysis, which will result in a requirements traceability matrix.

These capabilities support daily decision making, situational awareness, operations monitoring, rule-based processing and joint planning. Segment One will focus the interagency team in a proactive security environment and improve unity of effort with the following capabilities and supporting functions:

I. Integrate activities that support execution of business rules, data consolidation, information sharing, and workflow using automation to the greatest extent feasible.

a. Aggregate Tactical Information.
   i. Aggregate the information required to support the functions described below, such as interagency planning factors, weather conditions and forecasts, marine event schedules, harbor pilot schedules, and other key information types revealed by the domain analysis.
   ii. Aggregate CG enterprise information that relates to the situation picture, to include key systems such as NAIS, MAGNet, MISLE, COP.

b. Monitor Operations.
   i. Define and automate rulesets that apply:
      1. At all times.
      2. At key points in a vessel transit.
      3. As a result of alerts from other rules (e.g. only apply rule in MARSEC 2).
      4. As triggered by the watchstander on all vessels, categories of vessels, or individual vessels.
   ii. Monitor progress of execution of the Sector's operational plan of the day.
   iii. Manage the arrivals process.
      1. Automatically collect arrival data, e.g., Ships Arrival and Notification System (SANS) and Harbor Pilots (where available) related to ports within the Sector's Area of Responsibility (AOR). Ensure data changes are available to the system within 10 minutes (Objective: 5minutes) of an update from the reporting vessel.
      2. Automate USCG vessel arrivals processing to include aggregating vessel data and scoring results. Associate information with vessels appearing on sector arrivals list as a precursor to the joint planning review.
      3. Provide products for the joint planning process.
   iv. Ingest, process, correlate, and display track data and associate with relevant data objects (e.g., planned boardings) to include:
      1. National Automatic Identification System (NAIS)
      2. Common Operational Picture (COP)
      3. Port Partner Tracks where available
      4. Vessel Traffic Service (VTS)
      5. All available Blue Force Tracks, e.g., USCG, Customs and Border Protection (CBP), etc.
   v. Categorize vessels by size and type to support assessment and processing.
   vi. Identify and alert information anomalies to include at a minimum:
      1. Compare MAGNet, (Maritime Information System for Law Enforcement (MISLE), SANS, NAIS, and local data.
      2. Ingest alerts from MAGNet and trigger appropriate WatchKeeper actions.
      3. Vessel Score (ISPS): Alert generated ifvessel score threshold is reached and vessel requires boarding.
      4. Vessel Score (Port State Control (PSC) I, PSC II): Alert generated if vessel score threshold is reached and vessel requires boarding.

5. High Interest Vessel (HIV): Alert generated if vessel is identified as HIV during arrivals process (unclass elements only).
6. Vessel Not Scored Alert: Alert is generated if vessel has entered the situation picture and has not been scored/cleared by the Arrivals Desk.
7. SANS Arrival Updated/Need to Re-score: Alert generated if vessel has issued an updated Notice of Arrival (NOA).
8. Arriving and has No NOA : Alert generated if a vessel is in the AOR and has an arrival notice in Harbor Pilots system but no matching SANS NOA.
9. Early / Late Arrival Alert : Alert generated if a vessel enters the 6 mile boundary and her actual arrival time is +/- 6 hours of SANS arrival time.
10. OP Control Issued: Alert generated if a vessel has an Operational Control assigned.
11. Existing Watch Alerts (Manually set by users)

vii. Monitor vessel behavior throughout transit of critical or vulnerable areas. Vessel behavior should be monitored in relation to a set of rules resulting from a domain analysis. At a minimum these rules and actions should include:
   1. Individual Vessel Moves from Current Position: Alert generated if vessel movement watch is violated.
   2. Vessel enters/exits AOR zone: Alert generated if vessel movem nt watch is violated when the vessel enters OR exits an AOR zone. Users set enter/exit variable and zone variable - 6 mile, 12 mile, or 30 mile.
   3. Vessel enters/exits Port: Alert generated if vessel movement watch is violated by vessel entering OR exiting the port. Users set enter/exit variable.
   4. Vessel enters/exits anchorage : Alert generated if vessel movement watch is violated by vessel entering OR exiting the anchorage. Users set anchorage variable and enter/exit variable.
   5. Vessel enters/exits patrol area: Alert generated if vessel movement watch is violated by vessel entering OR exiting the patrol area. Users set patrol area variable and enter/exit variable.

c. Capture situational information (threshold: Search and Rescue (SAR), Law Enforcement (LE), Spills) received by watchstanders, automatically integrate with situation picture, and make easily accessible to Port Partners.
   i. Automate log keeping and generate official logs.
   ii. Automate Quick Response Cards (QRCs).
   iii. Capture manual entry of form data, define information object, represent object in operational picture, and generate appropriate alerts.

d. Generate notifications to Port Partners and other stakeholders that are triggered from processes such as interactive QRCs (iQRC).

e. Provide a capability to share a sensitive but unclassified situational picture with Interagency Operatons Center (IOC) partners that are physically present as well as external to the facility.

f. Provide ability for user to modify core (routine ops) rulesets and create and modify rulesets as appropriate to the situation. Provide interface that allows for simplified inclusion of any tactical elements.
   i. The system shall support the creation and management of rules that can employ all data types available within the system.
   ii. The system shall allow for certain rule sets to be employed only in specific geographic regions.
   iii. The interface must allow for the simplified inclusion of any tactical elements relevant to the rule and organize in a logical manner.

g. Automatically compile and generate standard reports (Threshold: standard reports, Objective : Generate new reports as emergent situations require.)
   i. Output to a variety of media, including: paper, email, electronic files, all compatible with interagency and CG enterprise-software for the standard workstation.
   ii. Output options must include: Text file with a compatible format for Coast Guard Message System; Display format optimized for screen viewing; other formats suitable as electronic attachments, automated notifications or inclusion in an email.

2. ***Support Joint planning for vessel arrivals and security activities among key interagency partners.***
    a. Automate the processes for coordinated and collaborative planning with partners both external to and/or present on the watchfloor.
    b. Support a joint operational planning process at a Sensitive But Unclassified (SBU) level.
        i. Automate the necessary collaboration tools to enable joint planning and targeting.
        ii. Conduct coordination meetings (e.g., "morning" meetings) and briefings with Port Partners both physically present in the center and in different locations.
            1. Include means for audio-visual communication.
            2. Include means to view briefing materials such as presentations, documents, and other views of tactical significance.
            3. Provide means to view WatchKeeper visualizations during the course of the briefing.
            4. Provide means for Port Partners to input required information to WatchKeeper.
        iii. Support the creation and maintenance of a Mission Request List and IOC Resource List.
            1. Mission Request List is comprised of non-emergent planning requests for CG and interagency assets and/or missions. Mission requests must capture all the required information necessary for the proper understanding, prioritization and scheduling of the mission. The system shall collect all future mission requests for use in the interagency planning cycle and in the development of the Weekly and Daily Operations Schedule.
            2. Capture the scheduled availability of interagency assets such as boats, aircraft, vehicles and people. Include at a minimum: asset names, status, operational schedule, dates available, contact information and parent command data.
    c. Ingest planning information from key interagency partners such as US Navy (USN), CBP, local pilots, and law enforcement to include: arrivals, scheduling, coordinated boarding, screening (including threat assessment) and resource availability.

3. ***Compose and maintain a situation picture.***
    a. Consume, organize, and display information for all missions conducted within the sector.
        i. Provide temporal and geospatial views of planned and emergent events that include outputs of the joint planning process and other situationally relevant information such as weather, maritime security status, force protection status, and asset status. These pictures provide high level "at a glance" views of the presence of information and cueing to significant changes in information content.
    b. Provide a capability to share a situational picture in an SBU environment.
    c. Provide means to assess information quality, relevance, and latency.
    d. Represent vessel!! with unique icons based on size, type, and category.
    e. Indicate vessel progress through the business rules (e.g., a "clear/not cleared" decision).

4. ***Assign resources to tasking.***
    a. Manage and allocate resources and tasking for routine and emergent situations such as law enforcement operations, inspections, and SAR as well as the results from the joint planning process.
    b. Create a task order that communicates Mission Request details to the assigned assets in formats that can be transmitted by the Coast Guard Message System, as well other electronic distribution.
        i. Output to a variety of media, including: paper, email, electronic files, all compatible with interagency and CG enterprise-software for the standard workstation. Output the details of task orders in a variety of media or formats, based on the user's selection.

ii. Output options must include: Text file with a compatible format for Coast Guard Message System; Display format optimized for screen viewing; other formats suitable as electronic attachments, automated notifications or inclusion in an email.

iii. Archive task requests when prepared such that the original tasking details are not overwritten during subsequent tasking modifications. Task modifications should stimulate a new task order while referencing the original.

c. Automate the interagency coordination, notifications, and workflow as a result of tasking, e.g., tasking statement.

d. Format schedules and calendars reflecting planned events, mission requests and asset assignments. Provide perpetual schedule with multiple views (day, week, month). Format schedules for screen display and other output media (paper, electronic, archival, etc).

Appendix A - Operational Scenarios

The following operational scenario and events are taken from the C21 PORD to illustrate contexts in which the operational functions provided in the Segment One description above may be executed.

Scene or Event	Possible Operational Functions
Sector receives Notice of Arrival for a commercial vessel arriving in 96-hours.	WatchKeeper adds the vessel to the Sector arrivals list that it automatically generates and maintains. WatchKeeper flags the new Arrival for processing by the Arrivals Desk. WatchKeeper creates an Arrivals Checklist for the vessel in preparation for screening by the Arrival Desk watchstander. The system automatically populates as much of the checklist as possible with information from a variety of enterprise and local information systems. WatchKeeper automatically consolidates arrival information and applies business rules for vessel arrivals. WatchKeeper automatically gathers the existing targeting information from CG and CBP and generates a consolidated arrival report as well as boarding recommendations. WatchKeeper finds track data in NAIS and links it with the Arrival.
Arrivals Desk completes screening for a commercial vessel arrival.	Views the Sector Arrivals List for unprocessed arrivals. Clears vessel data anomalies and schedules appropriate action (e.g. Cleared for Entry or Not Cleared/Boarding Required). User selects "schedule boarding" and the boarding request is routed through a routing slip (customizable at each sector to reflect different procedures). Approved boarding request is assigned to a Task Unit (TU), e.g. a small boat station or port partner asset, and forwarded. New tasking is forwarded to appropriate parties, to include the assigned units, the boarding team, port partners, any users who subscribe to these types of scheduling events. The system automatically solicits watclistander decisions in cases where data is ambiguous or inconclusive.
Sector receives report of oil sheen via landline.	Operations Controller opens Interactive Quick Response Card (iQRC) and selects event type (e.g. Pollution report, SAR, LE, etc). The iQRC provides prompts cueing the watchstander to gather complete information from the caller similar to a 9-1-1 call center system. The iQRC also prompts the watchstander with critical next steps based on the information entered and the sector business rules for the type of event. The system displays the sheen location reported on the iQRC and overlays potential dispersal on GIS Situation Picture. Reviews BFT locations and capabilities for appropriate response equipment and recommends appropriate responder. Allow the watchstander to select an area and screen for all recent transits (filter by time, vessel type). Automatically include event on various OPSUM views/reports. Notifications - prompt when National Response Center (NRC) notification thresholds are met. Notify required sector and port partner personnel as per established business rules. Track the status of the incident and integrate with the overall Sector status and situation picture. Creates and populates case-file information as required. Synchronizes the appropriate information with enterprise systems as needed.
Local patrol boat goes into "Charlie" status for maintenance.	Sector cutter notifies sector via message traffic that they assumed Charlie status for scheduled maintenance period. Watchstander updates the operational status for the vessel. The vessel status is updated on all views throughout system. The system compares the cutter schedule to the actual status and prompts watchstander of discrepancy between scheduled and actual status.
Change in arrival time for a vessel already processed by Sector arrivals desk.	The system receives updated arrival time from the local pilot's schedule. The arrivals desk screened the vessel 24-hours prior and assigned a boarding. The data change triggers notifications to the assigned units, personnel, the arrivals desk, and watchstanders.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California